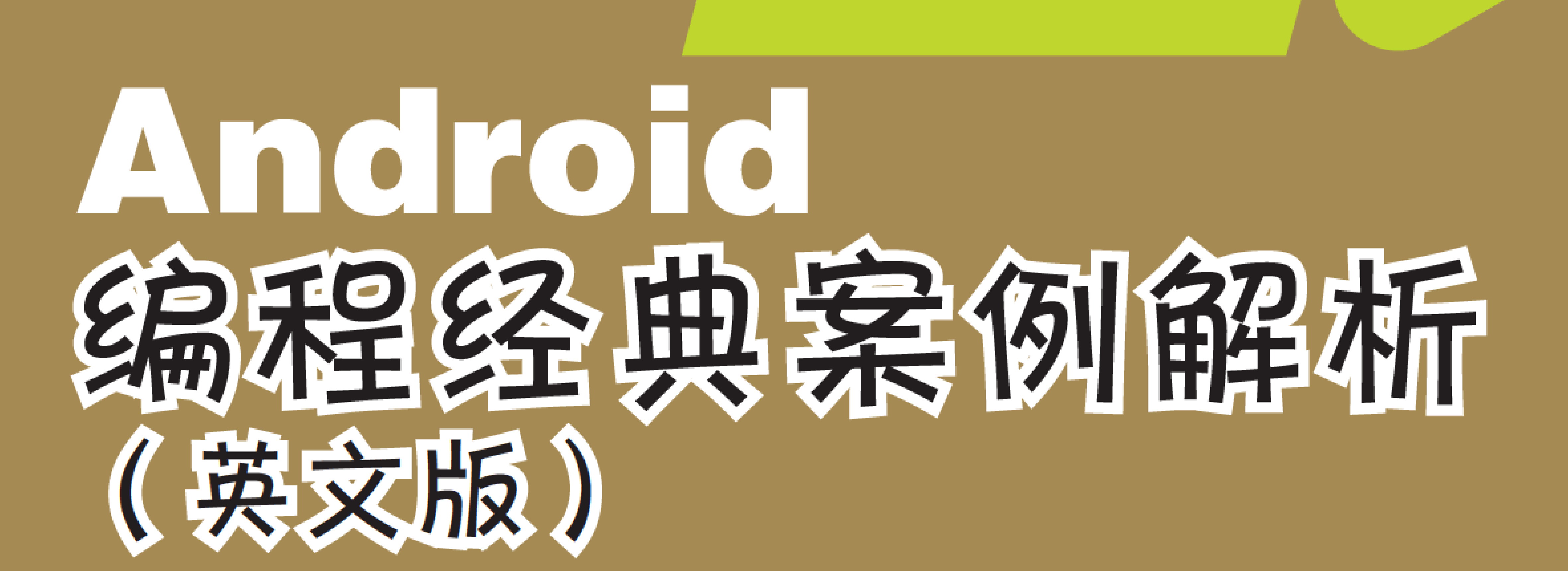


Android 编程经典案例解析（英文版）

Analysis of Classic Programming Cases in Android

钟元生 高成珍 等 编著

清华大学出版社

Android 编程经典案例解析
（英文版）
Analysis of Classic Programming Cases in Android

钟元生　高成珍　等 编著

清 华 大 学 出 版 社
北　　京

内 容 简 介

全书的 17 个 Android 编程案例，包括 TextView 特效、手机屏幕区域划分、我的课表、闪烁霓虹灯、简易计算器设计、页面滑动切换效果、图片定时滑动播放效果、搜索关键字提示、仿画廊视图效果、城市景点介绍、高校新闻等。为引导读者理解、掌握和灵活运用，编者通过图解分析、代码展示、技术剖析，由浅入深融会贯通。书中的例子稍加改动就可直接应用于实际的项目中。与本书配套的中文版教材经几千人试用，学习效果不错。

本书既可用作国内大学国际班的 Android 或 Java 类实践课程教材，也可供希望从事国际 APP 项目开发的程序员自学参考。掌握本书内容，计算机、软件工程专业的学生的就业机会将会大大增加。

图书在版编目（CIP）数据

Android 编程经典案例解析：英文 / 钟元生等编著. --北京：清华大学出版社，2016
ISBN 978-7-302-43409-2

Ⅰ. ①A…　Ⅱ. ①钟…　Ⅲ. ①移动终端 – 应用程序 – 程序设计 – 英文　Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字（2016）第 075238 号

责任编辑： 袁勤勇　李　晔
封面设计： 傅瑞学
责任校对： 时翠兰
责任印制： 杨　艳

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn, http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载： http://www.tup.com.cn,010-62795954
印 装 者： 三河市金元印装有限公司
经　　销： 全国新华书店
开　　本： 185mm×260mm　　**印　　张：** 16.75　　**字　　数：** 382 千字
版　　次： 2016 年 7 月第 1 版　　**印　　次：** 2016 年 7 月第 1 次印刷
印　　数： 1～1000
定　　价： 39.00 元

产品编号：066672-01

Preface by Authors

With the popularity of the Android phones and Android applications, the market demand for Android developer has grown rapidly. The lavish salary of Android programmer has attracted more and more students of colleges or universities to learn programming in Android. Many Chinese universities have now established a new course, Android Programming.

Therefore, we wrote a textbook, Android Application Development, based on the Android programming training handouts for helping college teachers in Jiangxi province to guide students to take part in the Mobile Application Software Design Contest. This book was published in January 2013 in Jiangxi College Press. The book was selected as textbook by many colleges and universities, such as Jiangxi Normal University, Jiangxi University of Finance and Economics, East China Institute of Technology University, Jiangxi Normal University of Science and Technology, University of Jinggangshan, Gannan Normal University, Jiujiang College; Jiangxi College of Applied Technology, Nanchang Institute of Technology, Jiangxi Vocational College of Environmental Engineering, etc. Besides Jiangxi province, there are several colleges in other provinces in China also selected this book as textbook of Android programming course, such as universities of Tianjin Sino-and-German Vocational and Technical College, School of Software at Xiamen University of Institute of Technology. At the same time, this book also has been selected as training book of Software College in Nanchang University, the Android training class at Nanchang Yiyou Company. The publication of this book, has promoted the process to start a new course of Android programming at some universities while attracted a number of users to study Android programming online.

Many teachers and learners think this book is very practical, suitable to learn how to program in Android vary from simple example to complex example step by step, suitable for self-study, easy to understand, especially suitable for teaching in classroom in the university. Many learners hope we could publish another book with commonly used cases which focus on the analysis of the function or effect that often used in actual Android App development to improve the reader's ability of developing Android App.

In the survey of job requirements related to Android programming, many enterprises have expressed the willing that hope to cooperate with us, so that we could help them to recruit Android App developers and test the skill of job applicants. Based on this, we have developed an online system for it and developed a set of architecture to test the user's programming skill in Android, including the primary, intermediate and advanced levels. In order to help the

examiner and examinee to clarify the contents and skills of this kind of test, we also provide some typical cases as a reference.

In order to improve student’s ability to develop App used in Android Phone, to test whether the student master the basic skills, to help companies to recruit the Android developers who had certain project experience and can immediately participate in the project, we had carefully divided the professional skills and basic knowledge in a rational way, eventually designed and programmed a textbook named “*Analysis of Classic Programming Cases in Android*” combined with our own teaching experience in Colleges or universities and the actual Android project development experience. These cases based on the original knowledge of Android programming and inserted some functions and other running effects into them, mainly inspect whether the student is able to master the programming method of Android, and whether they have self-learning ability. The design of these cases is mainly considered the following aspects.

(1) Practical. Simulate the common functions and effects in Android application development;

(2) Comprehensive. Each case involves a number of knowledge points, the student need to use them flexible;

(3) Pay attention to the case analysis. There are many Android program source codes on the Internet, but lack of the detailed analysis of development process. Besides, these codes had less comment and coding style is very different. These cases are not so easy to use in readers’ programming after downloaded them on the Internet that we pay attention to the detailed analysis of the case while designing and writing this book.

This book analyzes the development process of 17 typical Android cases in detail. Each case has higher practical value, and can be used in readers’ development projects with little modify by learner. At the same time, it introduces the common errors and the methods for debugging program in Android development. The corresponding Android quizzes are provided. After studying this book, readers will have the ability to develop Android application by themselves.

The writing work of this book is arranged as follows. Gao Chengzhen served as the chief editor, he is responsible for the case selection and the writing of most of the chapters, Zhong Yuansheng served as joint chief editor, specifically responsible for writing guidance, design style, compiles, peer review and quality assurance. Division of labor of each chapter are as follows: Gao Chengzhen is responsible for the Chapter 5th, 6th ,7th, 8th, 9th, 10th, 11th, 12th, 13th, 14th, 15th, 16th, 17th and 18th, Zhong Yuansheng is responsible for Chapter 1st , 2nd and 19th , Gao Bifan is responsible for the Chapter 3rd, and He Ying is responsible for the Chapter 4th. Yang Xu, Zhang Wen, Chen Haijun, Wu Weiwei, Huang Jing, Cao Quan and others joined in other works such as the draft discussion, editing the book and developing the supported teaching courseware.

During editing this book, we have got many helps and supports from many friends, such as leaders or experts from Software and Communication Engineering College at Jiangxi University of Finance and Economics, Computer Engineering Department at Jiangxi College of Science and Technology, Alliance Mobile Software Co., Ltd. in Nanchang, Cooperation Office between Colleges and Enterprises at Jiangxi Vocational College of Mechanical and Electrical Technology. Thanks all of them for their help.

Due to the limitation of our abilities, there may be some defects or errors in the book. We hope you can give us your valuable suggestion when you find out errors or unsuitable contents.

The authors of Chinese book
October 2014

Preface

Mobile application development has attracted more and more students to program in Android or IOS in these years all over the world. Some of them in China may work in the companies in USA, UA or other developed countries serving as an App programmer. In the meantime, there is huge demand among companies outside China to recruit App programmers. The demand cannot be satisfied often in the developed countries because there are no many students to pursuit degree with computer science relative subjects. Some of professors in the universities in the developed countries also want to recruit graduate students major in computer science relative subjects. Thus, some Chinese students proficient in English are recruited by these companies or universities.

However, many good App programmer especial male college students are not proficient in English under current situation in China. These students interested in programming are often very good at programming. According to some experiencing example of professors or foreign companies, the programmer can fulfil the programming task very good in the foreign companies although their English proficiencies is on the ordinary level.

With the popularity of Android mobile application, there are great gap between demand and supply in the marketplace of mobile programmer in the developed countries. Under current situation, many companies cannot find out this kind of Chinese programmer with good programming skills but English proficiency. Thus, there is low possibility to recruit them as a programmer. This situation is the same for the professors in universities to recruit graduate student for their research project with mobile applications.

On the other hand, there are many college students outside China with good English proficiency interested in China because there is great development chance in China these years. This trend will be more and clearer with the development of China for a long time. Those students can often understand few of Chinese words. The students hope to communicate with some Chinese students and set up some kind of friendship relationship. Although there are other ways to fulfill it, we think a new platform with new technology contest among students all around the world will be more suitable one to some person. So, we launch a new competition, named Mobile Application Development Contest. This idea has been communicated with many professor or experts outside China. Many positive responses give us encouragement to make it become true.

The contest provides a way to attract students from colleges and universities worldwide, to learn skills and knowledge how to program on mobile phones, and help companies developing mobile software to get in touch with best mobile software student programmers. It will also promote to interested companies original mobile software applications developed by college students.

Two “tracks” of competition are planned. They will be devoted to Android Application Programming Skills and Mobile Application Software Development.

The first track named Application Programming skill Contest (Android only). The aim of track is to test programming skills necessary to develop intelligent Android applications. The testing will consist of a 3-hour test results of which will be submitted to the contest server. The contestant’s score will consist of 40% for basic knowledge, and 60% for programming skills. The mark will be decided on the basis of correctness of submission and the total time of finishing the test. The answer will be blind reviewed by the experts designated by the Academic Committee.

The second track named Mobile Application Software Works Contest (No platform limitation, includes Android, iOS, or others). Applications developed for the competition must show creativity and practicality. There is no limit concerning the application area. The submission must include not only the executable program, but also the source code. Furthermore, the application design report, including PPT presentation and instruction manual must also be submitted. The complete set of files must be uploaded to the official competition website. Applications will be blind reviewed by the experts designated by the Academic Committee。

This book referenced another Chinese book authored by Gao Chengzhen, Zhong Yuansheng, He Ying and Gao Bifan published in January 2015. This book can use as a reference to this contest.

The translation work of this book is arranged as follows. Zhong Yuansheng are the head, he is responsible for the arrangement of translator, discussion of the translation style, translation quality control, overviewing and correcting the draft version, and the translation work of Chapter 1st, 2nd , 3rd and 4th. He also joints in the translation work of all other chapters and appendix. Gao Chengzhen is responsible for translation of the case code expecially the electronic version of each Android project case in this book and translation of all relative graphs in Chinese to version in English, and translation of Chapter 15th, 16th and 17th. He Ying is responsible for the translation of Chapter 7th , 8th and 10th. Gao Bifan is responsible for the translation of Chapter 9th , 12th and 13th . Huang Jing is responsible for the translation of Chapter 5th, 6th , 11th and 14th . Wu Weiwei is responsible for the translation of Chapter 17th and appendix. Zhao Shenglu, Chen Haijun, and Ding Yu are involved in other translation works such as the draft discussion, editing the book and developing the supported teaching

courseware.

Due to the limitation of our abilities, there may be some defects or errors in the book. We hope you can give us your valuable suggestion when you find out errors or unsuitable contents.

The authors

August 2015

Reader Guide

In this book, we assumed that you have got some basic knowledge of Android such as the structure of Android Application and some common views. If you have no knowledge about Android application developing, it is better for you to learn our textbook *Android Application Development Guide* or series of videos we recorded for teaching how to develop Android application step by step on Android 4.1. The resources website is http://www.XS360.cn/book/.

In this book, there are many cases; every case includes several source code files. We concerned on the program analysis of the example and only list some of the key code in the context in order to focus on the key point and introduce more knowledge in the limit pages. If you want to view all the codes, you can download these codes in our website. Importing the code into your IDE and run as android application, you can get the effect in the textbook.

It is strongly recommended that you complete the program by yourself based on the interpretation, description and the key code listed in the book while reading rather than directly run the case program to see the running result. Only when repeatedly failed several times, you can view the codes.

In order to facilitate teaching, line number for each section of source code is shown and some comments for some of key phrases are given out. Example source code is shown in the following.

```
1   public class MainActivity extends Activity {
2       public void onCreate(Bundle savedInstanceState) {
3           super.onCreate(savedInstanceState);
            // invoked the same method in parent class
4           setContentView(R.layout.activity_main);
            // setting the layout of the Activity
5       }
6       public boolean onCreateOptionsMenu(Menu menu) {
            //Creating Option Menu
7           getMenuInflater().inflate(R.menu.activity_main, menu);
            // specify the menu resource
8           return true;
9       }
10  }
```

In the code, 1, 2, 3… in the left is the line number, the “super.onCreate (savedInstanceState);”

in the middle is the real content of the program code. The symbol "//" and the following content "invoked the same method in parent class" indicates the comment of the middle code.

In order to facilitate learning, communication, resources sharing, we developed a website to download the appropriate resources including source code, courseware, papers and so on. The URL is: http://www.XS360.cn/ book/.

If you have any questions or any good suggestions during your study or reading the book, you can contact us by QQ group: 314753495 or email: 1281147324@qq.com.

Contents

Chapter 1 Special TextView Effects

1.1 Case Overview

This case mainly describes the special effects about the TextView, such as scrolling the text, setting a couple of text colors in the same TextView, displaying pictures around the text, and automatically identifying various links in the text. when our application running, it looks like Figure 1-1. (The TextView program is operating in Figure 1-1)

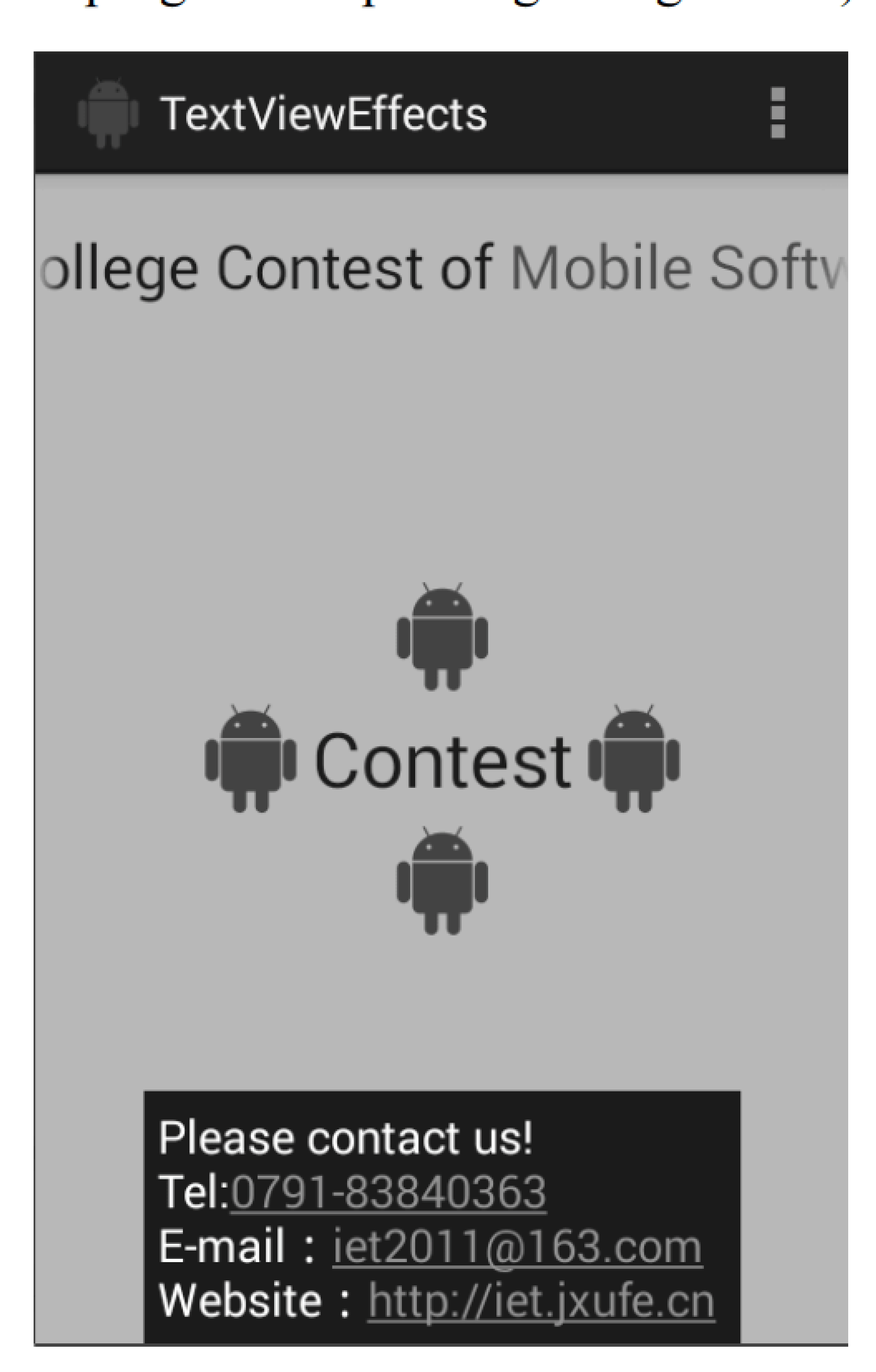


Figure 1-1 the figure of the running results

1.2 Key Code

Layout file: 01\TextViewEffect\res\layout\activity_main.xml

```
<FrameLayout xmlns:android= "http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc">
    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ellipsize="marquee"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:singleLine="true"
        android:textColor="#0000ff"
        android:textSize="24sp"
        android:layout_marginTop="20dp" />
    <TextView
        Android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center"
        android:textSize="30sp"
        android:text="@string/content"
        android:drawableTop="@drawable/ic_launcher"
        android:drawableBottom="@drawable/ic_launcher"
        android:drawableLeft="@drawable/ic_launcher"
        android:drawableRight="@drawable/ic_launcher"
        android:textColor="#0000ff"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/info"
        android:textColor="#ffffff"
        android:textSize="18sp"
        android:autoLink="all"
        android:layout_gravity="bottom|center_horizontal"
        android:background="#0000ff"
        android:padding="5dp" />
</FrameLayout>
```

String constants file: 01\TextViewEffect\res\values\strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name"> TextViewEffects </string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
```

```
    <string name="title"> Welcome to attend International College Contest of &lt;font color=red&gt;Mobile Software Development &lt;/font&gt;</string>
    <string name="content">Contest</string>
    <string name="info"> Please contact us!\nTel:0791-83840363
        \nE-mail: iet2011@163.com
        \nWebsite: http://iet.jxufe.cn</string>
</resources>
```

Program code: 01\TextViewEffect\src\iet\jxufe\cn\android\textvieweffect\MainActivity.java

```
public class MainActivity extends Activity {
    private TextView mTitle;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // find the widget in the layout file by the id
        mTitle=(TextView)findViewById(R.id.title);
        //set the content for TextView
    mTitle.setText(Html.fromHtml(getResources().getString(R.string.title)));
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //Inflate the menu; this adds items to the action bar if it is present
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

1.3 Code Analysis

1.3.1 The effect of scrolling text in the TextView

The effect of scrolling text, TextView needs to meet the following conditions:

A. The content of the TextView is longer (wider) than the text.

B. Set TextView as single-line display, **android:singleLine="true"**.[1] Otherwise, the word-wrap effect will be automatically applied to the text exceed the width specification by default.

[1] Note: Text words, short sentences or clauses in bold type in this textbook indicate that they are phrases of program codes or reserved keywork to distinguish them from other part.

C. Set the scrolling text display, **android:ellipsize="marquee"**, Otherwise the text exceed the width specification won't displayed after above setting.

D. TextView has focus, set attribute value **android: focusableInTouchMode** and **android: focusable** as true, when the TextView lose focus, the text will no longer be rolling.

1.3.2 Display various colors in the same text

In the TextView, we can set the color of the text by android: textColor. But this setting will change all words' color, not only the color of text. Using the android: textColor property, we cannot get what we need in the same text.

We can use Java code to realize the Muticolor effect. As we all know, Android well supported the Html and Html tags in the strings, which can be parsed by some static method in Html class, therefore what we need to do is just only setting the color of the text by Html tags in a string.

To set the content in code, firstly you need to get the corresponding widgets, which are uniquely specified by ID in Android. Therefore, you need to add the ID attribute to TextView widgets: android: id="@+id/title". You can easily obtain the widget by calling findViewById (R.id.title) and ultimately call the setText () method of the TextView to set the content.

The 10th line in MainActivity.java means: call the static method named “fromHtml”in Html class to parse a string. It mainly converts the Html tags into the corresponding display format. This string is corresponding resources of R.string.title ID, which is“Welcome to attend International College Contest of Mobile Software Development ”. < represent“<”, > represent“>”. Because character “<” and “>” have special meaning in the XML file, they can not be used directly in the string. The code can also be showed by:

```
mTitle.setText(Html.fromHtml("Welcome to International College Contest
of<font color=red> Mobile Software Development</font>"));
```

In the above codes, and are Html tags. The parse result is to set the text color between these tags into red.

1.3.3 Set picture Orientation

TextView can both display text and image in Android. The attributes android:drawable Top, android:drawableLeft specified the position of images, like located above, below, on the left or right of the text. It can also set the margins between the text and images. Through these we can easily achieve some simple effects with images and text. We put the TextView and ImageView together to create more complex effects.

Note: When the size of the image and the text are inconsistent, you can set the alignment

to obtain a better effect.

1.3.4 Automatic link

Links are frequently used in Android applications, particularly some promotional pages. We can call the corresponding program in the system to carry out some related operations after you click on the link, such as Web links, telephone links, email links. And it is very easy to implement these links in the Android, which can be achieved by setting the characteristics of android: auto Link attribute in TextView. The property values are listed as follow:

- None: Match no patterns (default).
- Web: Match Web URLs only, if there is web site in the text, the site will be displayed in the form of a hyperlink.
- Email: Match email addresses only, E-mail will be displayed in the form of a hyperlink.
- Phone: Match phone numbers only, phone number will be displayed in the form of a hyperlink.
- Map: Match map addresses.
- All: Match all patterns (equivalent to web|email|phone|map).

1.4 Expansion of Knowledge

1.4.1 The difference between android: gravity and android: layout_gravity

In the second **TextView** of the file **activity_main.XML** (line 20 and 21), we set **android: gravity = "center"** as well as **android: layout_gravity = "center"**. These two properties are used to set the way of alignment. So what is the difference between them?

The **android:gravity** indicated how the content of this widget is positioned within the widget itself. The **android:layout_gravity** indicated how this particular widget is positioned within its layout. The different effects of those two parts are shown in the Figure 1-2.

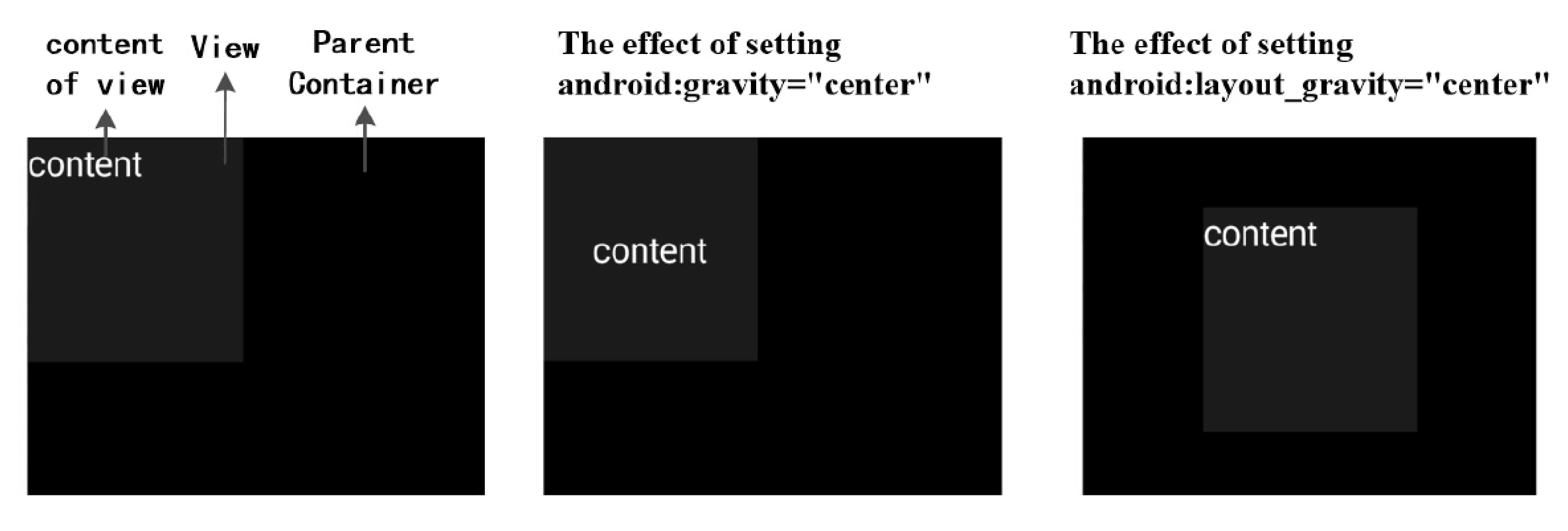


Figure 1-2 the figure of analysis of two kinds of alignment

1.4.2 The difference between android: padding and android: layout_margin

In the Android, the attributes of **android: padding** and **android: layout_margin** are both used to set the margin size. However, what's the difference between them?

Android: padding is the space inside the border between the border and the actual image or cell contents. Note that **padding** goes completely around the content: there is padding individually on the top, bottom, right and left sides (which can be independent), which means **padding** is a part of widget. The layout_margin is the space outside the border between the major container and the other elements next to this widget. **Padding** represents the distance inside of widgets. The margin represents the distance between widgets. The total effects are shown in the Figure 1-3.

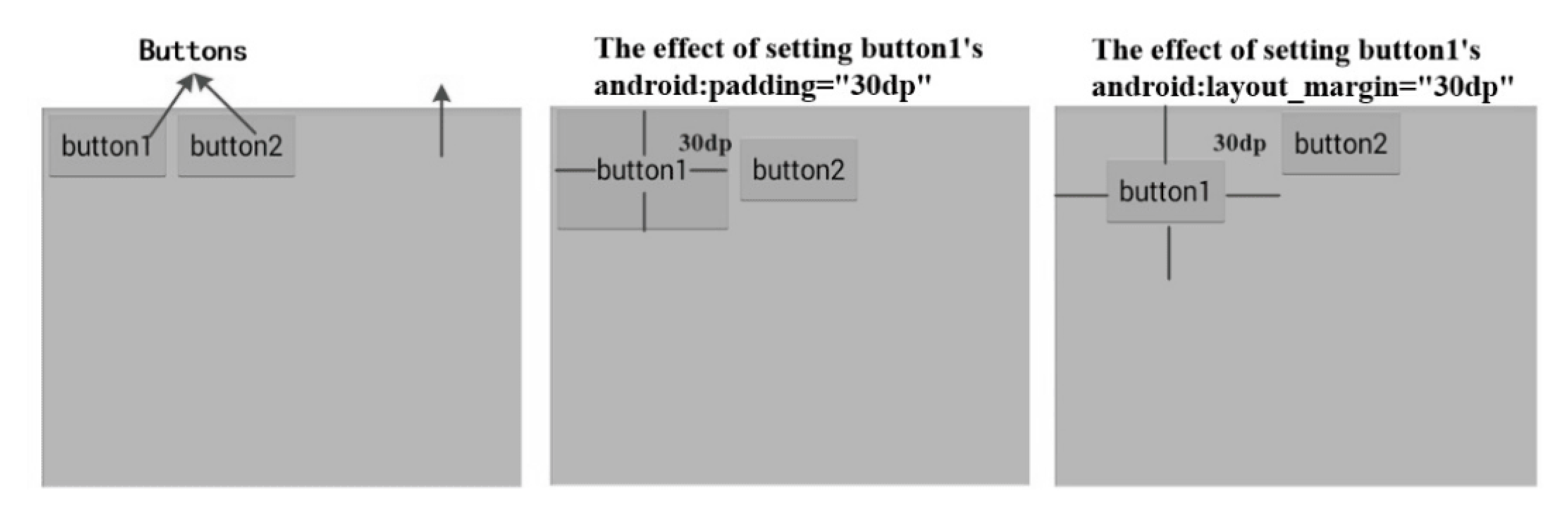


Figure 1-3 the figure of analysis of setting padding and margin

Note: Like the padding, the margin goes completely around the content: there are margins on the top, bottom, right, and left sides. If you only need to set the margin of one direction, you can use **android: paddingLeft** or **android: layout_marginLeft**.

1.4.3 The representation of color in Android

Color is widely used in Android, for example, set the color of the text and background in the widgets, etc. In the XML file, there are two ways to represent colors. One is represented through a hexadecimal number; another is represented based on some colors provided by the system. When it's represented by a hexadecimal number, the color values always begin with the # sign, followed by three primary colors as Red, Green or Blue, along with a transparency (Alpha) value. If you omit the transparency value, the color is completely opaque by defaults. Several specifications of color are listed as follow:

- **#RGB**: Use three hexadecimal numbers to represent color. R represents red, G represents green and B is blue. Each color has 16 levels which values from 0 to f;
- **#RRGGBB:** Use six hexadecimal numbers to represent color. RR represents red, GG represents green and BB is blue. Each color has 256 levels which values from 00 to ff;
- **#ARGB**: Use four hexadecimal numbers to represent color. A represents transparency ,

R represents red,G represents green and B is blue. Each color has 16 levels which values from 0 to f;

- **#AARRGGBB**: Use eight hexadecimal numbers to represent color, AA represents transparency, RR represents red, GG represents green and BB is blue. Each color has 256 levels which values from 00 to ff.

In Android XML file, the color is used like this:**@ android: color / color,** for example: **@ android: color / holo_red_dark** means deep red.

In addition, in the Android code, you can define a variety of colors in the **Color** class.

1.5 Thinking and Exercises

(1) **Framelayout** is used to implement the case in this chapter. Is it possible to achieve the same effect by using **LinearLayout**?

(2) Try to integrate the functions of **TextView** and **ImageView** to make images surrounded by text. **Tips**: use **RelativeLayout**.

(3) In Android, which of the following unit is recommended to set the size of the text(　).

A. px　　B. dp　　C. sp　　D. pt

(4) Which of the following options is not a value of color(　).

A. #ggg　　B. #ffff　　C. #eeeeee　　D. #dddddddd

Chapter 2 Phone Screen Division

2.1 Case Overview

Split screen is often used in practical application. For example, if the ratio of the width of the two widgets is 1:3, or keep one widget occupying 1/3 of the full screen, also as the reason of different sizes for phone screens, we cannot calculated the widget size by 1:3 or 1/3, then use the constant pixel values in the layout. We can easily reach the effect using Android: layout_ weight offered by the Android system. The case mainly introduces the usage of this attribute and splits the screen vertically into three sections, their height ratio is 1:4:1, and then spilt the middle part horizontally into three parts by the width ratio 1:4:1. Our application running looks like Figure 2-1.

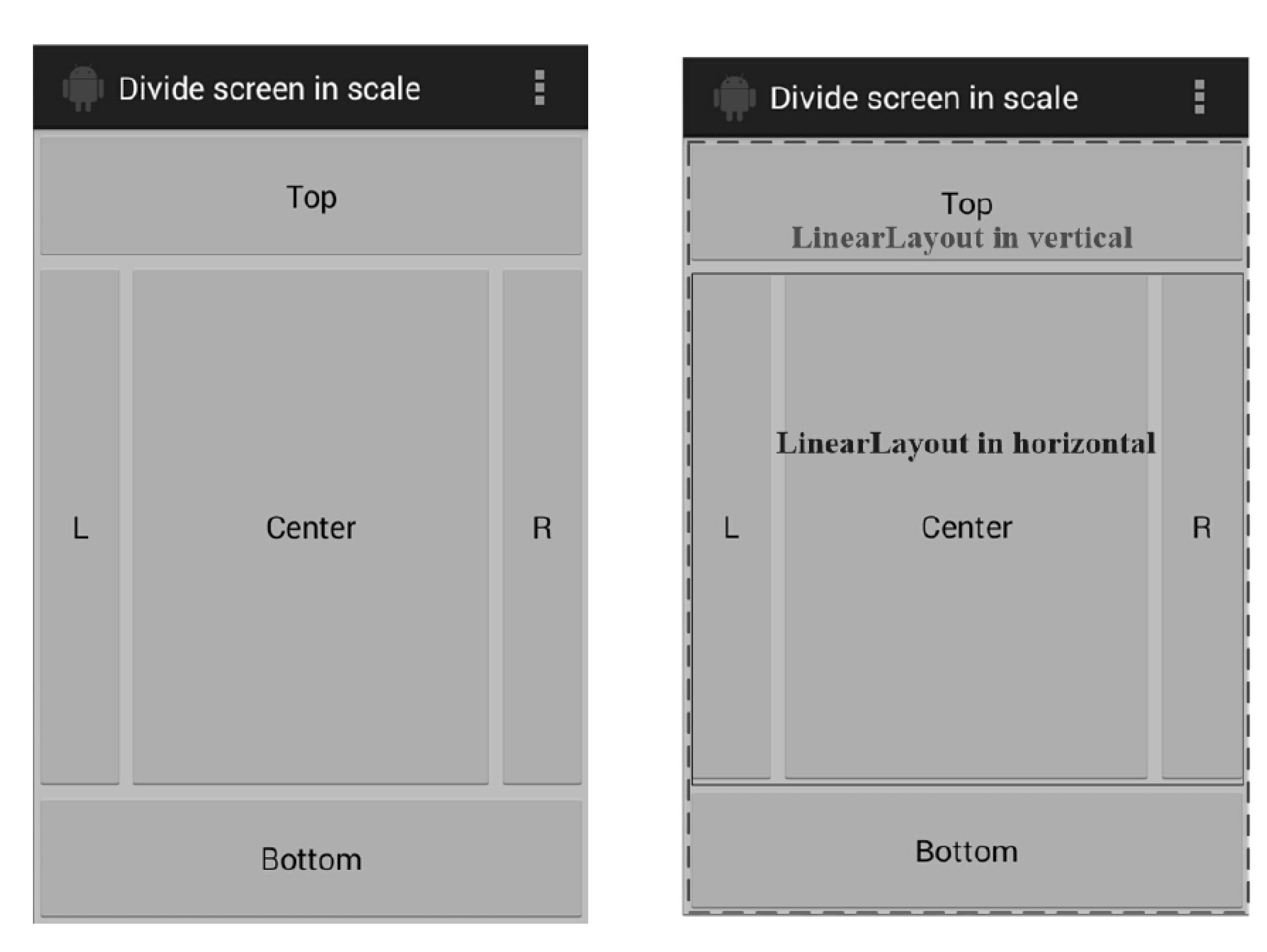


Figure 2-1 the figure of the running results and analysis

2.2 Key Code

Layout file: 02\DivideScreen\res\layout\activity_main.xml

```
<LinearLayout xmlns:android= "http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
     <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="@string/up"/>
     <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="4"
        android:orientation="horizontal">
        <Button
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="@string/left"/>
        <Button
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="4"
            android:text="@string/center"/>
        <Button
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="@string/right"/>
     </LinearLayout>
     <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="@string/down"/>
</LinearLayout>
```

String constant file: 02\DivideScreen\res\values\strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Divide screen in scale</string>
    <string name="action_settings">Settings</string>
    <string name="up">Top</string>
    <string name="down">Bottom</string>
```

```
    <string name="left">L</string>
    <string name="right">R</string>
    <string name="center">Center</string>
</resources>
```

2.3 Code Analysis

2.3.1 LinearLayout

LinearLayout is one of the most simplest and common layouts. The linear means that it can simply lay out one widget next to the other in one direction, either horizontally or vertically. When widgets have been laid out in one direction on the screen and there is no space left in this direction, at the same time, if some other widgets are going to be laid out on the same screen, those widgets would not be described. In other words, LinearLayouts will no change the directions due to represent the widgets on the screen. Common attributes in Linearlayout:

- **android:orientation**: Set the direction of LinearLayout, the options of its attributes are only vertical or horizontal. In which the default setting is horizontal.
- **android:gravity**: Specifies the alignment of widgets inside the LinearLayout, which can be applied on both the X and Y axes, within its own bounds. Must be one or more (separated by '|') constant values. For example, bottom|center_horizontal means the widget appears at the bottom of the screen and center horizontally.

2.3.2 Proportionally split screen

During the process of exploring Android applications, we usually prorated the widget size and even use a certain pixel in order to make the interface fit different mobile phone screen sizes. Android provides an attribute called **android:layout_weight** for widgets to represent the proportion of the widget in the free space, if there is only one widget set this attribute, the widget will fill the whole space. If there is multiple widgets have set in this attribute, the widgets will be allocated in proportion to the size of the extra space.

For example, there are three widgets e.g. a, b, c arranged horizontally and their relevant widths are wrapped content, X is the width of the entire screen. The layout weight value of the three widgets are: 1, 2, 3. It represents that the first widget account for the extra 1/6(6=1+2+3) of the remaining outer space width in additional to its own width. By parity of reasoning, the second widget account for the extra 1/3 of the remaining outer space width, the third account for the extra 1/2 respectively.

Remaining outer space = X – the width of a – the width of b – the width of c.

So the ration of the three widgets is (the width of a + 1/6 of the remaining outer space): (the width of b + 1/3 of the remaining outer space): (the width of c + 1/2 of the remaining outer

space), since the width of the widget is uncertain, we cannot get a certain width ratio between them to divide the screen by ratio. But there are two plans to reach the purpose:

(1) Set all the widget width at 0.

(2) Set all the widget width as match_parent.

When setting all the widgets' width at 0, the remaining outer space is X-0-0-0=X, and their width ratio is (0+1/6*X): (0+1/3*X): (0+1/2*X) = 1:2:3.

When setting all the widgets' width as match_parent, the remaining outer space is X-X-X-X=-2X, their width ratio is (X+1/6*(-2X)): (X+1/3*(-2X)): (X+1/2*(-2X)) = 2:1:0, which means widget c could not be displayed.

In summary, the best way to set the width of widget at a ratio is setting their width at 0, then setting their **android:layout_weight** as a certain proportion. Although we can also set their width as match_parent to achieve the purpose of dividing proportionally, the calculation is too much trouble.

Therefore, in this case, to divide the entire screen by the ratio of 1:4:1 vertically, we just set the widths of three widgets at 0 and the value of **android:layout_weight** as 1:4:1. To divide the middle section by the ratio of 1:4:1 horizontally, we set the width of the three widgets in the middle section at 0 and set the value of **android:layout_weight** as 1:4:1.

2.4 Extension of Knowledge

In practice, we often encounter a problem that it is difficult for a screen to display all the information, such as web information. We can add a scroll bar outside the widget and wrap the widget by the scroll bar when we need display the content beyond the screen indeed. There are two kinds of scroll bar: ScrollView and HorizontalScrollView, a scroll bar can only wrap a widget inside directly. For example, if you add 10 widgets in a horizontal LinearLayout, but actually it can only display up to 5 half according to the size of screen, since part of the 6th widget can be displayed, it will be compressed to display completely while the others cannot be displayed completely. If we need to show the four widgets beyond the screen, we can add a HorizontalScrollView for the horizontal LinearLayout.

2.5 Thinking and Exercises

(1) Can we use the attribute 'android:layout_weight' in FrameLayout? Why?

(2) Which of the following description about LinearLayout is correct (   )?

A. All the widgets in horizontal LinearLayout are displayed in accordance with the horizontal direction one by one. If beyond the width of the screen, it will automatically generate a horizontal scroll bar and you can drag the scroll bar to view other widgets

B. All the widgets in horizontal LinearLayout are displayed in accordance with the horizontal direction one by one. If beyond the width of the screen, the system will automatically wrap to display other widgets

C. All the widgets in horizontal LinearLayout are displayed in accordance with the horizontal direction one by one. If beyond the width of the screen, the redundant widgets will not be displayed

D. All the widgets in horizontal LinearLayout are displayed in accordance with the horizontal direction one by one. If beyond the width of the screen, continue add widgets, runtime error

(3) By which attribute can we set to make the width of widgets become a certain proportion in LinearLayout (　)?

A. android:layout_width　　B. android:layout_weight

C. android: layout_margin　　D. android:layout_gravity

(4) How many sub widget can a ScrollView wrap directly (　)?

A. 0　　B. 1　　C. 2　　D. Unlimited

Chapter 3 My Course Table-TableLayout

3.1 Case Overview

This case mainly introduced the usage of TableLayout, we can easily align a group of widgets to the left,right, top,or bottom,and easily make the width of widgets the same through TableLayout. It is very useful for some straight interfaces (for example, rows and columns). This case is aimed to realize the student's schedule. More specifically, one student have 7 different classes every day, thus, the schedule is a typical table that arranged in rows and columns. Our application running looks like Figure 3-1.

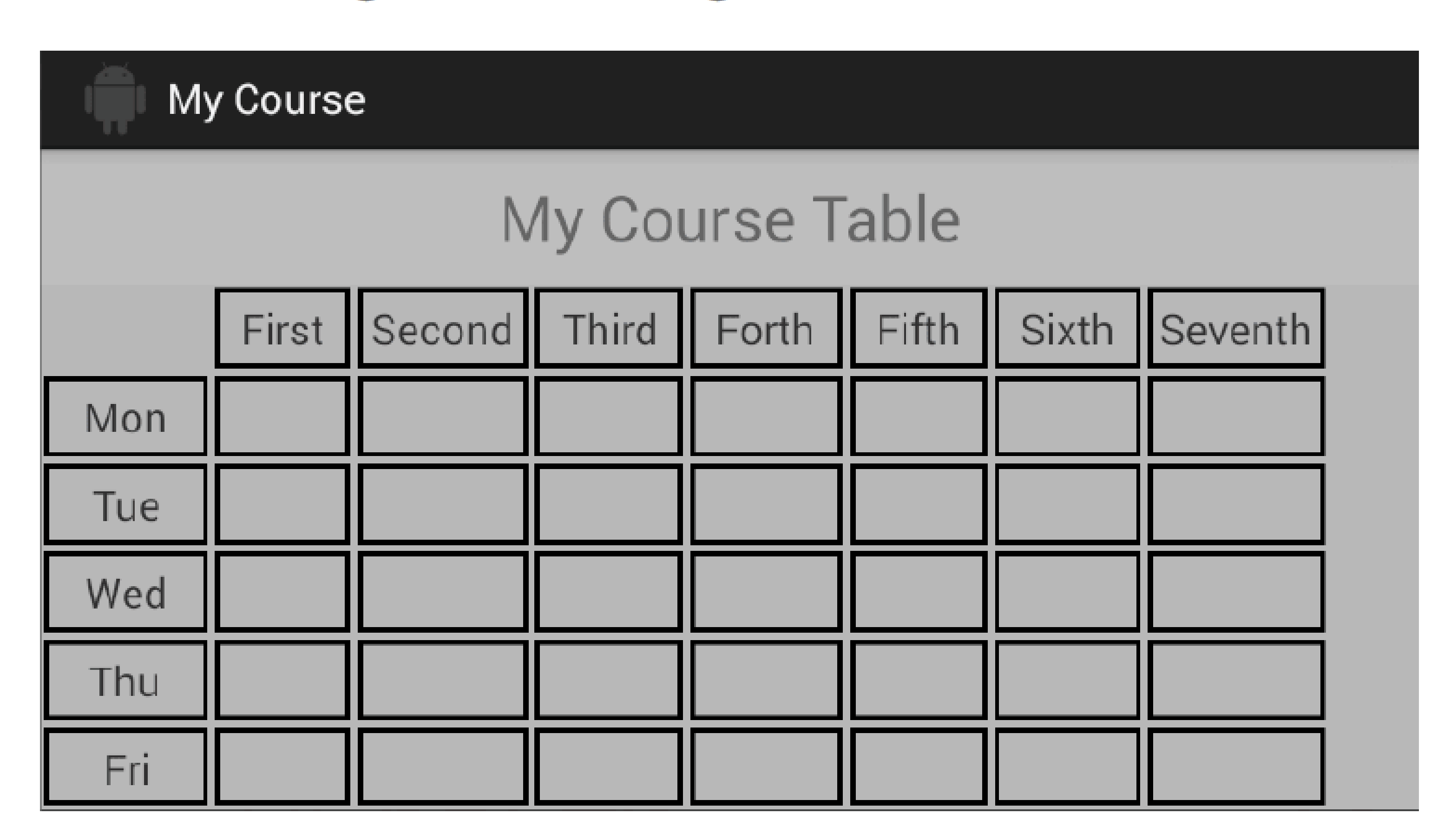


Figure 3-1 the figure of the running results

3.2 Key Code

Layout file: 03\CourseList\res\layout\activity_main.xml

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc" >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
```

```
            android:text="@string/mycur"
            android:textColor="#ff2233"
            android:background="#ccbbaa"
            android:padding="10dp"
            android:textSize="24sp" />
        <TableRow>
            <TextView
                style="@style/textView"
                android:layout_column="1"
                android:text="@string/first" />
            <TextView
                style="@style/textView"
                Android:text="@string/second" />
            <TextView
                style="@style/textView"
                android:text="@string/third" />
            <TextView
                style="@style/textView"
                android:text="@string/forth" />
            <TextView
                style="@style/textView"
                android:text="@string/fifth" />
            <TextView
                style="@style/textView"
                android:text="@string/sixth" />
            <TextView
                style="@style/textView"
                android:text="@string/seventh" />
        </TableRow>
        <TableRow>
            <TextView
                style="@style/textView"
                android:text="@string/mon" />
            <TextView style="@style/textView" />
            <TextView style="@style/textView" />
            <TextView style="@style/textView" />
            <TextView style="@style/textView" />
            <TextView style="@style/textView" />
            <TextView style="@style/textView" />
            <TextView style="@style/textView" />
        </TableRow>
        <TableRow>
            <TextView
                style="@style/textView"
```

```
            android:text="@string/tue" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
    </TableRow>
     <TableRow>
        <TextView
            style="@style/textView"
            android:text="@string/wed" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
    </TableRow>
    <TableRow>
        <TextView
            style="@style/textView"
            android:text="@string/thu" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
    </TableRow>
    <TableRow>
        <TextView
            style="@style/textView"
            android:text="@string/fri" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
        <TextView style="@style/textView" />
```

```
    </TableRow>
</TableLayout>
```

String constants file: 03\CourseList\res\values\strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My Course</string>
    <string name="mon">Mon</string>
    <string name="tue">Tue</string>
    <string name="wed">Wed</string>
    <string name="thu">Thu</string>
    <string name="fri">Fri</string>
    <string name="first">First</string>
    <string name="second">Second</string>
    <string name="third">Third</string>
    <string name="forth">Forth</string>
    <string name="fifth">Fifth</string>
    <string name="sixth">Sixth</string>
    <string name="seventh">Seventh</string>
    <string name="mycur">My Course Table</string>
</resources>
```

Style file: 03\CourseList\res\values\styles.xml

```
<resources xmlns:android="http://schemas.android.com/apk/res/android">
    <style name="AppBaseTheme" parent="android:Theme.Light"></style>
    <style name="AppTheme" parent="AppBaseTheme"></style>
    <style name="textView">
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:layout_margin">1dp</item>
        <item name="android:gravity">center</item>
        <item name="android:textSize">16sp</item>
        <item name="android:background">@drawable/bg</item>
    </style>
</resources>
```

3.3 Code Analysis

3.3.1 Class schedule interface analysis

It is clear that there are 48 TextView widgets in the interface. The TextView widget displayed “My Course Table” with its own row (see Figure 3-2). A table of 6 rows and 8columns is under the TextView widget, but the first row and first column are both empty. The

width and height of each TextView widgets are the same.Every TextView widgets with its own border. In order to reuse codes, you can define the styles for them individually.

My Course

My Course Table a view in the row

Blank	First	Second	Third	Forth	Fifth	Sixth	Seventh
Mon							
Tue							
Wed		6 rows	and 8	columns			
Thu							
Fri							

Figure 3-2 the figure of analysis of the running results

3.3.2 TableLayout

TableLayout lays out its widgets in the form of a table and consists of only other TableRow widgets. TableLayout represents a row in a table and can contain other UI widgets. TableRow widgets are laid out next to each other horizontally,sort of like LinearLayout with a horizontal orientation.

In the TableLayout, the width of each column is the same, the width of the column is decided by the widest widget. The width of the TableLayout depends on the width of the primary container. By default, it is always filled with primary container .

The main attributes and explainations of TableLayout are described below:

- **android:collapseColumns**: Hide the specified column, Its value for the column where the serial number is started from 0. If you need to hide multiple columns, commas will separate numbers.
- **android:shrinkColumns**: Shrink the specified column, so that the entire line can not exceed the full-screen, which is used when the content of a row over the width of the screen. At this time, it makes the specified column compression wrap, its value for the column where the serial number. If you don't specify this propert, beyond the part of the screen will automatically intercept, does not show。
- **android:stretchColumns**: Expand the specified column to fill the blank part of the screen. When the line is not enough content to fill the entire screen, this property is used to fill the entire screen. At this time, specifiing the width of the column will be expanded to fill the blank part，The width of the other columns is unchanging. Forms can contain multiple lines,the number of columns per row may be different, but the width of the same row is the same, will not because there is a row to spare, making expansion the width of a column in the row, other lines don't have any spare, and make

them the column width is constant.

- **android:layout_column**: The columns of the specified widget in TableRow. If not setting this property, by default, widgets in a row will be an arranged one next to the other one. By setting this property, it can specify a widget's column, so as to achieve the middle one is empty.
- **android:layout_span**: Specify a number of columns spanned by the widgets, will soon be consolidated into a multi-column column.

If you want to realize the effect the title has its own row, you just put the TextView widget into the Tablelayout.

If you want to realize the effect, the first row and column will be empty, just setting the attribute of first widget in the first row like this :android:layout_column="1".

3.3.3 Add borders to TextView

In the Android, TextView widgets is not able to set the border attribute. How can we add borders to TextView widgets? There are mainly three methods: a.You define a custom widget ,and this widget inherit the TextView widget, then override the onDraw() method, you can get the border; b.Set a picture which background color is transparent ,then you get the border; c.Define an XML file, and use it as background of the TextView widget .

Here we using the third method, we define a rectangle, the width of its border is 2dp and the color is black; while the middle of the rectangle is transparent, then set the background of rectangle into the TextView.Specific codes is shown as follow:

Custom graphics files: 03\CourseList\res\drawable-hdpi\bg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >
    <solid android:color="#00000000" />
    <padding android:left="5dp"
        android:top="5dp"
        android:right="5dp"
        android:bottom="5dp"/>
    <stroke
        android:width="2dp"
        android:color="#000000" />
</shape>
```

3.3.4 Definition of style

In the schedule interface,there are a lot of TextView widgets, all of them need to set the width, height, the size of text, the background color, etc. Most TextView widgets are alike. If

you set them one by one, there will be a lot of work, especially when you want to change the style of these widgets, you had to modify each of them. It's a troublesome work. We can set the attribute together and define it as the style, if the widgets need to use the attribute, the widgets can just use the style. So like this, we can just modify one attribute, that all the widgets changed at the same time, it's quite convenient. If the user just need modify one widget's attribute, he can set a new attribute in the widget and assign to it, which will cover the properties in style.

We usually define the style file in the < resource > root element, add < style > element, specify the style name by the name attribute. Using <item> labels to show the attributes in the style; Specify the specific properties by the name attribute in the <item> labels; the content of the<item> label is the value of the attributes. When referenced, just set style attribute value like this :@ **style/style name**. Specific code is listed in the styles.xml.

3.4 Expansion of Knowledge

In Android,mobile, the phone usually has vertical screen, sometimes, horizontal screen can displayed more beautiful, the most common is to play video, and generally the video is 16:9 or 4:3, when using vertical screen to play it, which is too small and less attractive. The Android system provides two methods to convert the vertical screen into horizontal screen, as shown below:

a. Add a **<android: screenOrientation = "landscape"/>** element within the **<manifest>** block in the corresponding activity which need to be played in horizontal screen.

b. Have a judge in the code, If it is a vertical screen, then set it to the horizontal screen, code shown below:

```
if(getRequestedOrientation()!=ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE){
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
}
```

3.5 Thinking and Exercises

(1) In this example Table Layout just can be filled on the screen, If the student has 10 classed a day, how to change in order to check the 10 classes?(Suggest to add the scroll bar)

(2) Which of the following is the right way to define a custom style?( )

A. <resources>

<style name="myStyle">

<item name="android:layout_width">match_parent</item>

</style>

</resources>

B. <style name="myStyle">

<item name="android:layout_width"> match_parent </item>

</style>

C. <resources>

<item name="android:layout_width"> match_parent </item>

</resources>

D. <resources>

<style name="android:layout_width"> match_parent </style>

</resources>

(3) Which of the following description about TableLayout is not correct? (　)

A. TableLayout inherited from LinearLayout

B. Tablelayout can be clearly specified, it includes how many rows and columns

C. In TableLayout, you can set a widget　account for multiple columns

D. If you add a widget directly into the TableLayout, not adding TableRow widget, then the widget will has its own row

(4) Which of the following option is the right way to set a column as a retractable column in TableLayout? (　)

A. Set the properties of TableLayout: android:stretchColumns="x", x Represents the column number

B. Set the properties of TableLayout: android: shrinkColumns="x", x Represents the column number

C. Set properties for concrete columns: android:stretchable="true"

D. Set properties for concrete columns: android: shrinkable="true"

Chapter 4　Images Around Text—RelativeLayout

4.1　Case Overview

This case mainly introduced the usages of RelativeLayout, we can easily place a widget up/down/left/right to the particular widget and make them alignments by using RelativeLayout. In the following case, it put RelativeLayout, TextView widget and ImageView widget together to realize the effect images around text. Our application running looks like Figure 4-1.

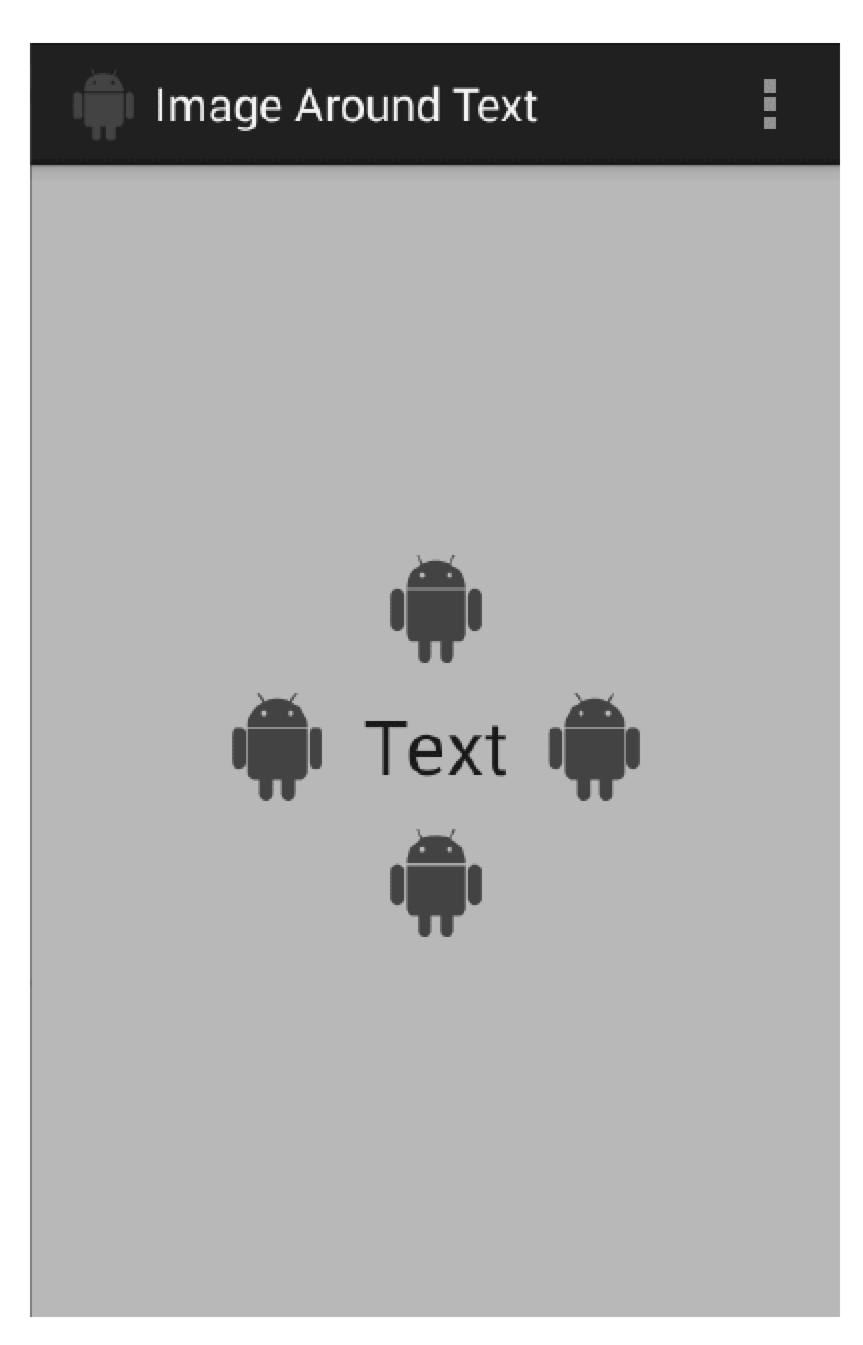


Figure 4-1　the figure of the running results

4.2　Key Code

Layout file: 04\ImageAroundText\res\layout\activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:background="#aabbcc" >
    <TextView
        android:id="@+id/center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:padding="10dp"
        android:text="@string/text"
        android:textColor="#0000ff"
        android:textSize="30sp" />
    <ImageView
        style="@style/myImgStyle"
        android:layout_centerVertical="true"
        android:layout_toLeftOf="@id/center"
        android:contentDescription="@string/imageInfo" />
    <ImageView
        style="@style/myImgStyle"
        android:layout_centerVertical="true"
        android:layout_toRightOf="@id/center"
        android:contentDescription="@string/imageInfo" />
    <ImageView
        style="@style/myImgStyle"
        android:layout_above="@id/center"
        android:layout_centerHorizontal="true"
        android:contentDescription="@string/imageInfo" />
    <ImageView
        style="@style/myImgStyle"
        android:layout_below="@id/center"
        android:layout_centerHorizontal="true"
        android:contentDescription="@string/imageInfo" />
</RelativeLayout>
```

String constant file: 004\ImageAroundText\res\values\strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Image Around Text</string>
    <string name="action_settings">Settings</string>
    <string name="text">Text</string>
    <string name="imageInfo">Image</string>
</resources>
```

Add these codes in the label <resource> in ImageAroundText \res\values\styles.xml

```
<style name="myImgStyle">
<item name="Android:layout_width">wrap_content</item>
```

```
<item name="Android:layout_height">wrap_content</item>
<item name="Android:src">@drawable/ic_launcher</item>
</style>
```

4.3 Code Analysis

4.3.1 Interface analysis

It is clear that there are five widgets in the interface, they are one TextView widget and four ImageView widgets; In which the TextView widget is located in the center of screen and four ImageView widgets respectively located up/down/left/right to the TextView widget in alignment with the center TextView widget.

If we use the LinearLayout to realize the effect, then we need to nest a horizontal LinearLayout into a vertical LinearLayout, it will generate a lot of codes, and behaving sluggishly. This is a simple case, we also can use TableLayout to realize it, but considering its weak extensibility, it is more reasonable for us to use LinearLayout.

4.3.2 RelativeLayout

RelativeLayout lays out its children relative to each other. Having said that RelativeLayout requires each of its child views to have an ID set so that we can position it relative to other children.

There are two kinds of references in RelativeLayout, one is called primary container the current RelativeLayout, the other refers to a particular widget. There is only one primary container, so when the widget position is relative to the primary container or they have alignment relations, its valid values is true or false.

The main attributes and explanations of RelativeLayout are described below:

- **android:layout_centerHorizontal**: Rule that centers the child horizontally with respect to the bounds of its RelativeLayout parent.
- **android:layout_centerVertical**: Rule that centers the child vertically with respect to the bounds of its RelativeLayout parent.
- **android:layout_centerInParent**: Rule that centers the child with respect to the bounds of its RelativeLayout parent.
- **android:layout_alignParentTop**: Rule that aligns the child's top edge with its RelativeLayout parent's top edge.
- **android:layout_alignParentBottom**: Rule that aligns the child's bottom edge with its RelativeLayout parent's bottom edge.
- **android:layout_ alignParentLeft**: Rule that aligns the child's left edge with its RelativeLayout parent's left edge.

- **android:layout_ alignParentRight**: Rule that aligns the child's right edge with its RelativeLayout parent's right edge.
- **android:layout_toRightOf**: Rule that aligns a child's left edge with another child's right edge.
- **android:layout_toLeftOf**: Rule that aligns a child's right edge with another child's left edge.
- **android:layout_above**: Rule that aligns a child's bottom edge with another child's top edge.
- **android:layout_below**: Rule that aligns a child's top edge with another child's bottom edge.
- **android:layout_alignTop**: Rule that aligns a child's top edge with another child's top edge.
- **android:layout_ alignBottom**: Rule that aligns a child's bottom edge with another child's bottom edge.
- **android:layout_ alignLeft**: Rule that aligns a child's left edge with another child's left edge.
- **android:layout_ alignRight**: Rule that aligns a child's right edge with another child's right edge.

Usually we need two aspects of information to place a widget: the position and alignment.

Note: An inside widget can be a reference,an outside widget can not use as a reference.

4.4 Extension of Knowledge

In the example 01, it seems like more easily to place the images around the text through **android: drawable XXX** method of TextView widget. But it has the limitation that only four images can be added from up, down, left, right directions around the text.

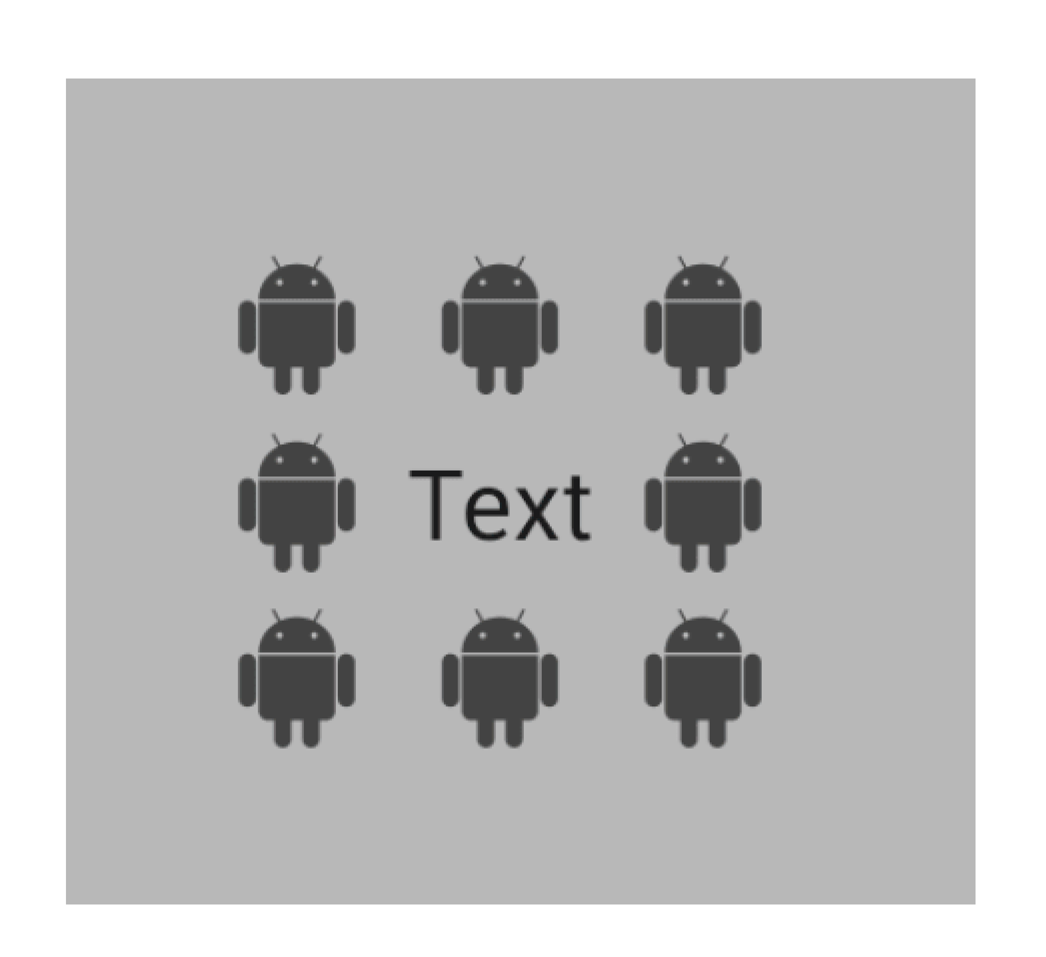


Figure 4-2 the figure of the running results

If you want to realize the effect adding images from eight directions as shown in Figure 4-2, it can be realized by adding some Image View widgets via RelativeLayout. Firstly, using an ID set for the four widgets: up, down, left, right, and then adding four ImageView widgets. Specific code is listed as follow:

```
<ImageView
   style="@style/myImgStyle"
   android:layout_alignTop="@id/up"
```

```
    android:layout_alignLeft="@id/left"
    android:contentDescription="@string/imageInfo" />
<ImageView
    style="@style/myImgStyle"
    android:layout_alignTop="@id/up"
    android:layout_alignLeft="@id/right"
    android:contentDescription="@string/imageInfo" />
<ImageView
    style="@style/myImgStyle"
    android:layout_alignBottom="@id/down"
    android:layout_alignLeft="@id/left"
    android:contentDescription="@string/imageInfo" />
<ImageView
    style="@style/myImgStyle"
    android:layout_alignBottom="@id/down"
    android:layout_alignLeft="@id/right"
    android:contentDescription="@string/imageInfo" />
```

In actual, we use different methods to meet different requirements. If simply adding images to a certain position of the text, TextView widget is enough to charge it. But if the situation is more complex and require more, RelativeLayout is a better choice.

4.5 Thinking and Practice

(1) Using TableLayout to realize the above two effects.

(2) In RelativeLayout,if you want to place a widget at the center of screen,you can set(　).

A. android:gravity="center"　　B. android:layout_gravity="center"

C. android:layout_centerInParent="true"　　D. android:scaleType="center"

(3) In RelativeLayout,which attribute can only be true or false(　).

A. android:layout_alignTop　　B. android:layout_alignParentTop

C. android:layout_toLeftOf　　D. android:layout_above

Chapter 5 Flashing Neon—FrameLayout

5.1 Case Overview

This case mainly introduced the usages of FrameLayout; we can use FrameLayout to realize the effect, the superposition of multiple widgets. This example is combined with Timer and Handler messaging mechanism to achieve the effect like flashing neon. Using Timer to send the message at regular time, after the handler received the message and then changed the background color of the widgets to realize a twinkling effect. Our application running looks like Figure 5-1.

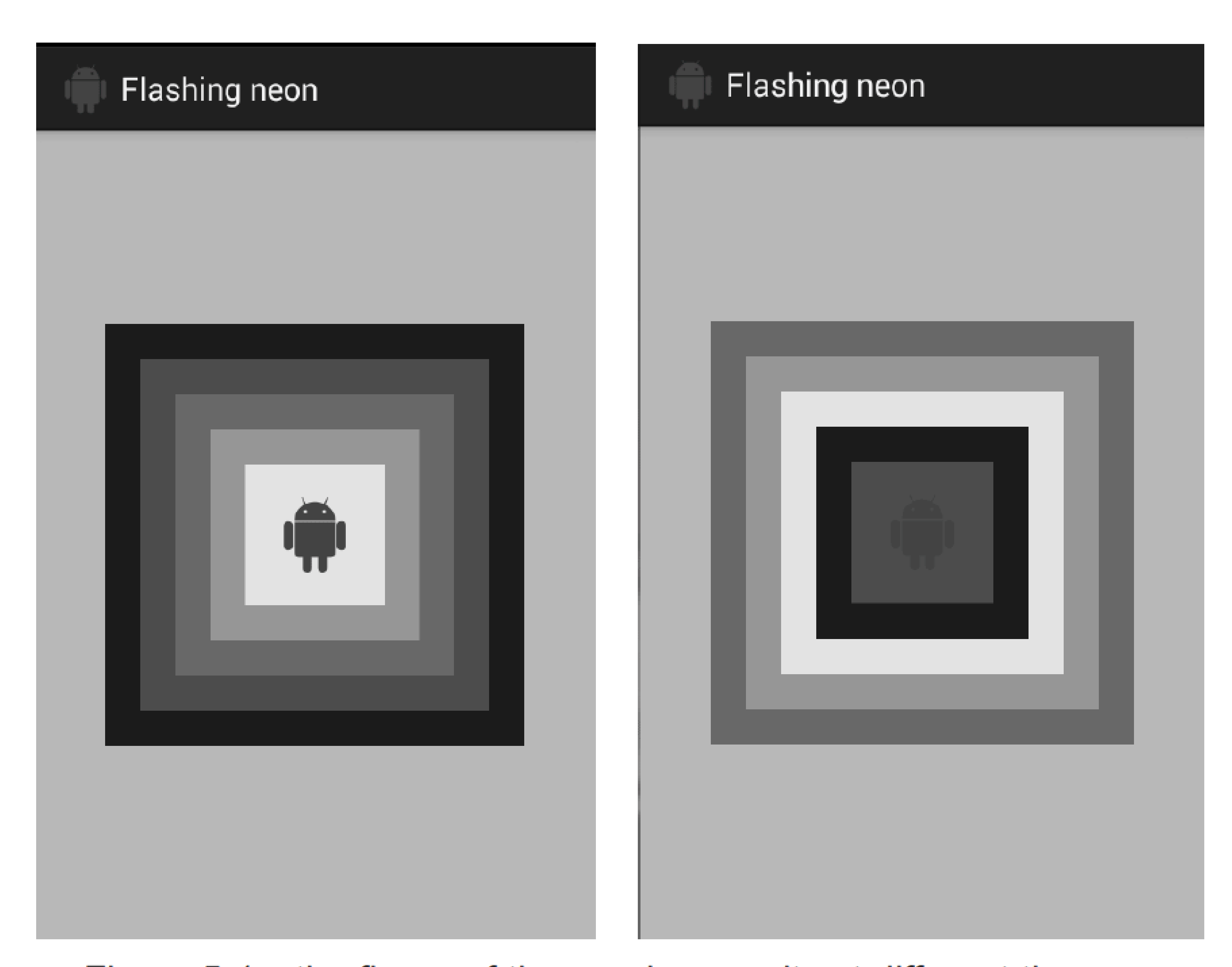


Figure 5-1 the figure of the running results at different times

5.2 Key Code

Layout file: 05\FrameLayoutTest\res\layout\activity_main.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/
android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc">
     <TextView
        android:id="@+id/text01"
        android:layout_width="240dp"
        android:layout_height="240dp"
        android:layout_gravity="center"/>
     <TextView
        android:id="@+id/text02"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_gravity="center"/>
     <TextView
        android:id="@+id/text03"
        android:layout_width="160dp"
        android:layout_height="160dp"
        android:layout_gravity="center"/>
     <TextView
        android:id="@+id/text04"
        android:layout_width="120dp"
        android:layout_height="120dp"
        android:layout_gravity="center"/>
     <TextView
        android:id="@+id/text05"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:layout_gravity="center"/>
     <ImageView
        android:src="@drawable/ic_launcher"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:contentDescription="@string/imgInfo"/>
</FrameLayout>
```

The string constant file is easy.

Program code:

05\FrameLayoutTest\src\iet\jxufe\cn\Android\framelayouttest\MainActivity.java

```
public class MainActivity extends Activity {
    //Define an array used to store the id of the TextViews
    private int[] textIds = new int[] { R.id.text01, R.id.text02,
```

```
                                R.id.text03, R.id.text04, R.id.text05 };
    //Define an array used to store colors
    private int[] colors = new int[] { Color.RED, Color.MAGENTA,
                                Color.GREEN, Color.YELLOW, Color.BLUE };
    //Define an array used to store TextViews, the length of array equals
the length of textIds
    private TextView[] views = new TextView[textIds.length];
    private Handler mHandler;
    private int current = 0;//record the index of current TextView
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Access the element of textIds and get the TextView by the value
          of element
        for (int i = 0; i < textIds.length; i++) {
            views[i] = (TextView) findViewById(textIds[i]);
        }
        //Create a handler object used for receive and process message
        mHandler = new Handler() {
            //The method which process message
            public void handleMessage(Message msg) {
                if (msg.what == 0x11) {//Judge whether the message is
                                the specific message
                    //set the background color of textview in circle
                    for (int i = 0; i < views.length; i++) {
                        views[i].setBackgroundColor(colors[(i + current)
                                % colors.length]);
                    }
                    current = (current + 1) % colors.length;
                }
            }
        };
        Timer timer = new Timer();//Create a timer object
        //start the TimerTask,send a message in 3000 milliseconds
        timer.schedule(new TimerTask() {
            public void run() {
                mHandler.sendEmptyMessage(0x11);
            }
        }, 0, 3000);
    }
}
```

5.3 Code Analysis

5.3.1 Interface analysis

It is clear that there are six widgets in the interface, they are one ImageView widget and five TextView widgets; in which the ImageView is located in the center of screen and it displays the icon of this application. The five TextView widgets also located in the center of screen, in which the lowest TextView widget is the biggest, the size of other TextView widgets decrease in proper order. The background color of these TextView widgets will be dynamicly changed, through XML that cannot realize this effect, we need to add ID for these TextView widgets, and then find the relevant widgets by the ID.

According to the characteristics of this interface, it is a complex effect, use LinearLayout or TableLayout can not realize this effect. We can use RelativeLayout to realize it, put all the widgets located in the center of screen, then add the widgets into the relative layout according to superpose order. This case FrameLayout seems to be a better choice.

5.3.2 FrameLayout

FrameLayout places its children on top of each other so that the latest child is covering the previous, like a deck of cards. This layout policy is useful for tabs, for example,FrameLayout is also used as a placeholder for other widgets that will be added programmatically at some later point in time. We can use attribute **android:layout_gravity** to set the location. Through Frame Layout , we can realize the effect widgets superpose together.

Note: Each widget has a frame exclusively, which means each of them do not have any relationship with others, you can not split screen according to the proportion in FrameLayout.

5.3.3 The timer

In this case, we realize the effect of changing the background color of widgets dynamically, which is also periodically changed. This performance can be achieved by starting a thread, then starting an endless loop in the loop body, every loop cycling, the thread will sleep 3000milliseconds, then we realize the effect that every 3000 milliseconds the color of background changed. Android has better encapsulation, which can provide timer class. When you need to perform periodic operation, just create a timer object, then call the schedule() method, this method will pass three parameters, the first parameter is the time task object, it shows the specific perform operation. The timer task is an abstract class, it includes abstract method run(), the timer class can not be instantiated, it must create a subclass of timer class to realize the run()method; the second parameter is the delay time which keep track of perform

operation, the unit is millisecond; the third parameter is the period time that perform operation, the unit is millisecond. The program code from 34th to 38th lines means every 3000 milliseconds send a message.

Why not directly to change the background color of widgets in the run() method? Why use handler? Because in Android, the widget is not thread-safe. None thread safe means if multiple threads perform operation on a widget, the result could be different. To avoid this situation, Android has a clearly regulation that all the operation of interface can only be placed in the main thread, instead of operating in the child thread. While we did not see the child thread in the program, but the timer class actually created a child thread. The run() method did not run in the main thread, so we can not change the background color in the run() method.

5.3.4 Handler message passing

The main thread can make changes in interface, however it did not know when to change, the child thread wants to change the interface, but it can not change, it is a contradiction. At this time, we need to use a certain intermediately to let the main thread interact with child thread. That's how handler message passing mechanism comes out in Android. The Handler class mainly includes the following several methods in Table 5-1.

Table 5-1 main methods for Handler class

Methods Name	Function
public void handleMessage (Message msg)	Subclasses must implement this to receive messages
public final boolean sendEmptyMessage (int what)	Sends a Message containing only the what value
public final boolean sendMessage (Message msg)	Pushes a message onto the end of the message queue after all pending messages before the current time
public final boolean hasMessages (int what)	Check if there are any pending posts of messages with code 'what' in the message queue
public final boolean post (Runnable r)	Causes the Runnable r to be added to the message queue

From these methods, we can see the handler is mainly used in sending message, receiving message, processing message. The execution process: when we need to perform an operation on interface, we use Handler to send message in the child thread. Once the message is sent successfully, it will call the handleMessage (Message msg) in handler class. The method is in the main thread, so it can change the interface. The handler message mechanism can be summarized as who sends it, who processes it, when to send a message and the message is going to be processing automatically.

The handleMessage(Message msg)method is a callback method, when handler received message, system will be called automatically. Therefore, it usually needs to override the method when creating a handler object. The program codes from 20th line to 30th line showed

that you can write the relevant business logic in the handleMessage(Message msg) method. A handler object can send multiple messages, so we need to judge the category of message when we received the messages, then take different treatment to different messages.

5.4 Extension of Knowledge

When you need to dynamically change the statement information of widgets via java codes, you need to add ID for the widget in the interface, then use findViewById() method to control the widget in codes. If there are more widgets need to be dynamically changed, the code will be a lot. So, the codes can be simplified according to the actual situation. In this case, there are five widgets in the interface, the operations of them are alike, so we can respectively put the widgets and the IDs of them into two arrays, then we can get the IDs through the for loop, according to the IDs we can find the related widgets in the loop body, then assigned them to the corresponding widgets in the arrays. When we need to change the background color of all the widgets, just processing a “for loop”.

Using this method to simplify the codes need to meet the certain criteria:a.the widgets’ type are alike, and there are a lot of widgets. b.the operation of all widgets are similar. c.operations of widgets meet certain regularity.

5.5 Thinking and Practice

(1) Please using RelativeLayout to realize this effect.

(2) In this case we use timer class to realize the effect that dynamically change the background color, please using normal thread to realize the effect.

(3) There is a button in FrameLayout, if we want it to be located in the center of screen, how to set the attribute? (　)

A. set the button’s attribute: android:layout_gravity="center"

B. set the button’s attribute: android:gravity="center"

C. set the parent container’s attribute: android:layout_gravity="center"

D. set the parent container’s attribute: android:gravity="center"

Chapter 6　Design Calculator—Use Multiple Layout

6.1　Case Overview

This case designed and realized an easy interface of calculator. We can use different ways to represent the effect, either through various layouts, like LinearLayout, TableLayout, Relative Layout, or GridLayout which first published in Android4.0. This example also introduced the usage of the Android style. At this point, our application looks like Figure 6-1 when running.

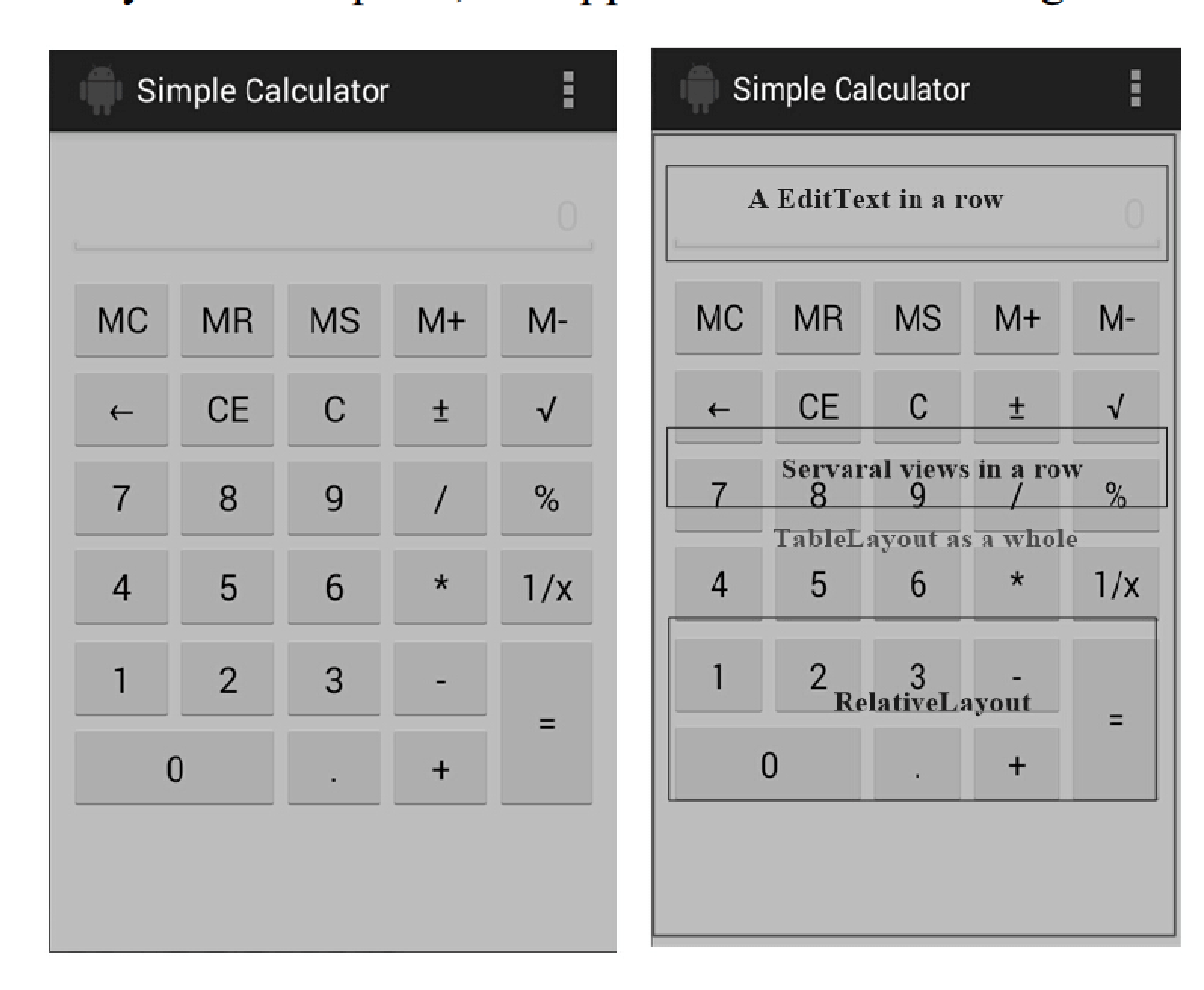


Figure 6-1　the figure of the running results and analysis

6.2　Key Code

Layout file: 06\CalculateTest\res\layout\activity_main.xml

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <EditText
        android:layout_width="match_parent"
```

```
        android:layout_height="60dp"
        android:layout_margin="10dp"
        android:enabled="false"
        android:gravity="right|bottom"
        android:inputType="numberDecimal"
        android:textSize="24sp"
        android:text="@string/zero" />
    <TableRow android:gravity="center_horizontal">
        <Button
            style="@style/myStyle"
            android:text="@string/mc" />
        <Button
            style="@style/myStyle"
            android:text="@string/mr" />
        <Button
            style="@style/myStyle"
            android:text="@string/ms" />
        <Button
            style="@style/myStyle"
            android:text="@string/mplus" />
        <Button
            style="@style/myStyle"
            android:text="@string/mminus" />
    </TableRow>
    <TableRow   android:gravity="center_horizontal">
        <Button
            style="@style/myStyle"
            android:text="@string/arrow" />
        <Button
            style="@style/myStyle"
            android:text="@string/ce" />
        <Button
            style="@style/myStyle"
            android:text="@string/c" />
        <Button
            style="@style/myStyle"
            android:text="@string/plusminus" />
        <Button
            style="@style/myStyle"
            android:text="@string/correct" />
    </TableRow>
    <TableRow android:gravity="center_horizontal">
        <Button
            style="@style/myStyle"
```

```
            android:text="@string/seven" />
        <Button
            style="@style/myStyle"
            android:text="@string/eight" />
        <Button
            style="@style/myStyle"
            android:text="@string/nine" />
        <Button
            style="@style/myStyle"
            android:text="@string/div" />
        <Button
            style="@style/myStyle"
            android:text="@string/mode" />
    </TableRow>
    <TableRow android:gravity="center_horizontal">
        <Button
            style="@style/myStyle"
            android:text="@string/four" />
        <Button
            style="@style/myStyle"
            android:text="@string/five" />
        <Button
            style="@style/myStyle"
            android:text="@string/six" />
        <Button
            style="@style/myStyle"
            android:text="@string/mul" />
        <Button
            style="@style/myStyle"
            android:text="@string/daoshu" />
    </TableRow>
    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal" >
        <Button
            android:id="@+id/one"
            style="@style/myStyle"
            android:layout_marginTop="2dp"
            android:text="@string/one" />
        <Button
            android:id="@+id/two"
            style="@style/myStyle"
            android:layout_alignTop="@id/one"
```

```
            android:layout_toRightOf="@id/one"
            android:text="@string/two" />
        <Button
            android:id="@+id/three"
            style="@style/myStyle"
            android:layout_alignTop="@id/one"
            android:layout_toRightOf="@id/two"
            android:text="@string/three" />
        <Button
            android:id="@+id/minus"
            style="@style/myStyle"
            android:layout_alignTop="@id/one"
            android:layout_toRightOf="@id/three"
            android:text="@string/minus" />
        <Button
            android:id="@+id/equal"
            style="@style/myStyle"
            android:layout_height="100dp"
            android:layout_alignTop="@id/one"
            android:layout_toRightOf="@id/minus"
            android:text="@string/equal" />
        <Button
            android:id="@+id/plus"
            style="@style/myStyle"
            android:layout_alignBottom="@id/equal"
            android:layout_toLeftOf="@id/equal"
            android:text="@string/plus" />
        <Button
            android:id="@+id/dot"
            style="@style/myStyle"
            android:layout_alignBottom="@id/equal"
            android:layout_toLeftOf="@id/plus"
            android:text="@string/dot" />
        <Button
            style="@style/myStyle"
            android:layout_width="120dp"
            android:layout_alignBottom="@id/equal"
            android:layout_toLeftOf="@id/dot"
            android:text="@string/zero" />
    </RelativeLayout>
</TableLayout>
```

String constant file: 06\CalculateTest\res \values\strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
    <string name="app_name"> Simple Calculator </string>
    <string name="action_settings">Settings</string>
     <string name="mc">MC</string>
    <string name="mr">MR</string>
    <string name="ms">MS</string>
    <string name="mplus">M+</string>
    <string name="mminus">M-</string>
    <string name="arrow">←</string>
    <string name="ce">CE</string>
    <string name="c">C</string>
    <string name="plusminus">±</string>
    <string name="correct">√</string>
    <string name="zero">0</string>
    <string name="one">1</string>
    <string name="two">2</string>
    <string name="three">3</string>
    <string name="four">4</string>
    <string name="five">5</string>
    <string name="six">6</string>
    <string name="seven">7</string>
    <string name="eight">8</string>
    <string name="nine">9</string>
    <string name="plus">+</string>
    <string name="minus">-</string>
    <string name="mul">*</string>
    <string name="div">/</string>
    <string name="mode">%</string>
    <string name="equal">=</string>
    <string name="dot">.</string>
    <string name="daoshu">1/x</string>
</resources>
```

Add these codes in label <resource> in CalculateTest\res\values\styles.xml

```
<style name=" myStyle">
    <item name="android:layout_width">60dp</item>
    <item name="android:layout_height">50dp</item>
    <item name="android:textSize">20sp</item>
    <item name="android:gravity">center</item>
</style>
```

6.3 Code Analysis

6.3.1 Interface analysis

It is clear that there are one EditText and 28 Buttons in the calculator interface, the width

of the EditText is filling the parent container and the content in the EditText can not be modified by users, and the content is changed according to the user's operation. Among the 28 Buttons, 26 Buttons' style is alike except two others. The height of "=" Button is twice as the normal Button, and the width of "0"is twice as the normal Button.

For most Buttons, they have its own rules, the whole is located by the rules, and we can consider using TableLayout. However, TableLayout only allows widget over the column, does not allow widget cross row. So only using TableLayout cannot realize the effect, we need to use other layout nested in TableLayout. According to the last two rows, the RealativeLayout is the best choice, we use Table Layout inside the RelativeLayout.

Note:When using TableLayout, if the widget is not located in the label <TableRow>, the widget will has its single line; When using RelativeLayout, the reference can only exist as an internal widget inside, it can not be the widget outside.

6.3.2 Define style

In the calculate interface, there are a lot of widgets, each of them need to set the width, height, the size of text, the align style, etc. The styles of most Buttons are alike. If you set the Buttons one by one, it will be a lot of work, there will be so many codes, especially when you want to change the style of these Buttons, you had to modify each of them, which is a troublesome work. We can set the attribute together and define it as the style, if the widgets need to use the attribute, the widgets can just use the style. So like this, we can just modify one attribute, that all the widgets will be changed at the same time, which is quite convenient. If the user just needs to modify one widget's attribute, he can set a new attribute in the widget and assign to it, this will cover the properties in style.

6.4 Extension of Knowledge

GridLayout

In Android4.0, GridLayout is the new coming layout; it has the advantages LinearLayout, TableLayout and RelativeLayout. The GridLayout contains a rowSpec and a columnSpec, which divided the whole container into rows and columns, and each grid can place a widget. What's more, you can set a widget across columns or across rows, and control the direction of placement in rows or in columns. The main attributes of GridLayout:

- **android:rowCount**: The maximum number of rows to create when automatically positioning children.
- **android:columnCount**: The maximum number of columns to create when automatically positioning children.
- **android:orientation**: The orientation that widgets lay out in GridLayout the value is

vertical or horioutal and if not set the attribute horizontal is default.

- **android:layout_row**: set the widget's row number.
- **android:layout_rowSpan**: set the widget across the count of rows.
- **android:layout_column**: set the widget's column number.
- **android:layout_columnSpan**: set the widget across the count of columns.

specific codes as follows:

Layout file: 06\CalculateTest\res\layout\gridlayout.xml

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:columnCount="5">
    <EditText
        android:layout_height="60dp"
        android:layout_columnSpan="5"
        android:layout_margin="10dp"
        android:enabled="false"
        android:gravity="right|bottom"
        android:inputType="numberDecimal"
        android:textSize="24sp"
        android:text="@string/zero"
        android:layout_gravity="fill_horizontal"/>
    <Button
        style="@style/myStyle"
        android:text="@string/mc"/>
    <Button
        style="@style/myStyle"
        android:text="@string/mr"/>
    <Button
        style="@style/myStyle"
        android:text="@string/ms"/>
    <Button
        style="@style/myStyle"
        android:text="@string/mplus"/>
    <Button
        style="@style/myStyle"
        android:text="@string/minus"/>
    <Button
        style="@style/myStyle"
        android:text="@string/arrow"/>
    <Button
```

```
        style="@style/myStyle"
        android:text="@string/ce"/>
    <Button
        style="@style/myStyle"
        android:text="@string/c"/>
    <Button
        style="@style/myStyle"
        android:text="@string/plusminus"/>
    <Button
        style="@style/myStyle"
        android:text="@string/correct"/>
    <Button
        style="@style/myStyle"
        android:text="@string/seven"/>
    <Button
        style="@style/myStyle"
        android:text="@string/eight"/>
    <Button
        style="@style/myStyle"
        android:text="@string/nine"/>
    <Button
        style="@style/myStyle"
        android:text="@string/div"/>
    <Button
        style="@style/myStyle"
        android:text="@string/mode"/>
    <Button
        style="@style/myStyle"
        android:text="@string/four"/>
    <Button
        style="@style/myStyle"
        android:text="@string/five"/>
    <Button
        style="@style/myStyle"
        android:text="@string/six"/>
    <Button
        style="@style/myStyle"
        android:text="@string/mul"/>
    <Button
        style="@style/myStyle"
        android:text="@string/daoshu"/>
    <Button
        style="@style/myStyle"
        android:text="@string/one"/>
```

```
    <Button
        style="@style/myStyle"
        android:text="@string/two"/>
    <Button
        style="@style/myStyle"
        android:text="@string/three"/>
    <Button
        style="@style/myStyle"
        android:text="@string/minus"/>
    <Button
        style="@style/myStyle"
        android:text="@string/equal"
        android:layout_rowSpan="2"
        android:layout_height="100dp"/>
    <Button
        style="@style/myStyle"
        android:text="@string/zero"
        android:layout_columnSpan="2"
        android:layout_width="120dp"/>
    <Button
        style="@style/myStyle"
        android:text="@string/dot"/>
    <Button
        style="@style/myStyle"
        android:text="@string/plus"/>
</GridLayout>
```

In MainActivity delete the setContentView (R.layout.activity_main); add this **set ContentView(R.layout.grid layout);**

Note: The GridLayout is the new coming layout in Android4.0, so we need to set the minimum version in the AndroidManifest.xml, **android:minSdkVersion = "14"**.

6.5 Thinking and Practice

(1) In this example, we have used TableLayout combined with RelativeLayout to realize the calculator's interface; can you use a combination of other layout to realize it again? Try it by yourself.

(2) In order to place the content horizontally in the center of each row, set the android:gravity="center_horizontal" in each <TableRow> element, can we set the android:gravity="center_horizontal" in the <TableLayout>, to realize the same effect? why or why not ? think it .

(3) Which one is not a layout in the Android? (　)

A. FrameLayout　B. GridLayout　C. BorderLayout　D. TableLayout

Chapter 7 Page Slide Show

7.1 Case Overview

This case mainly introduced the effect of page slide. Putting several key pages together in the same application through ViewPager, you can easily switch to the previous or the next page by sliding the screen or clicking on the title left and right. Meanwhile, small icons, which stand for the page number, will change at the same time at the bottom of screen. The red one means currently selected page, while the yellow one means the page unselected. When you click a certain icon, you can also switch to the corresponding page, our application running looks like Figure 7-1.

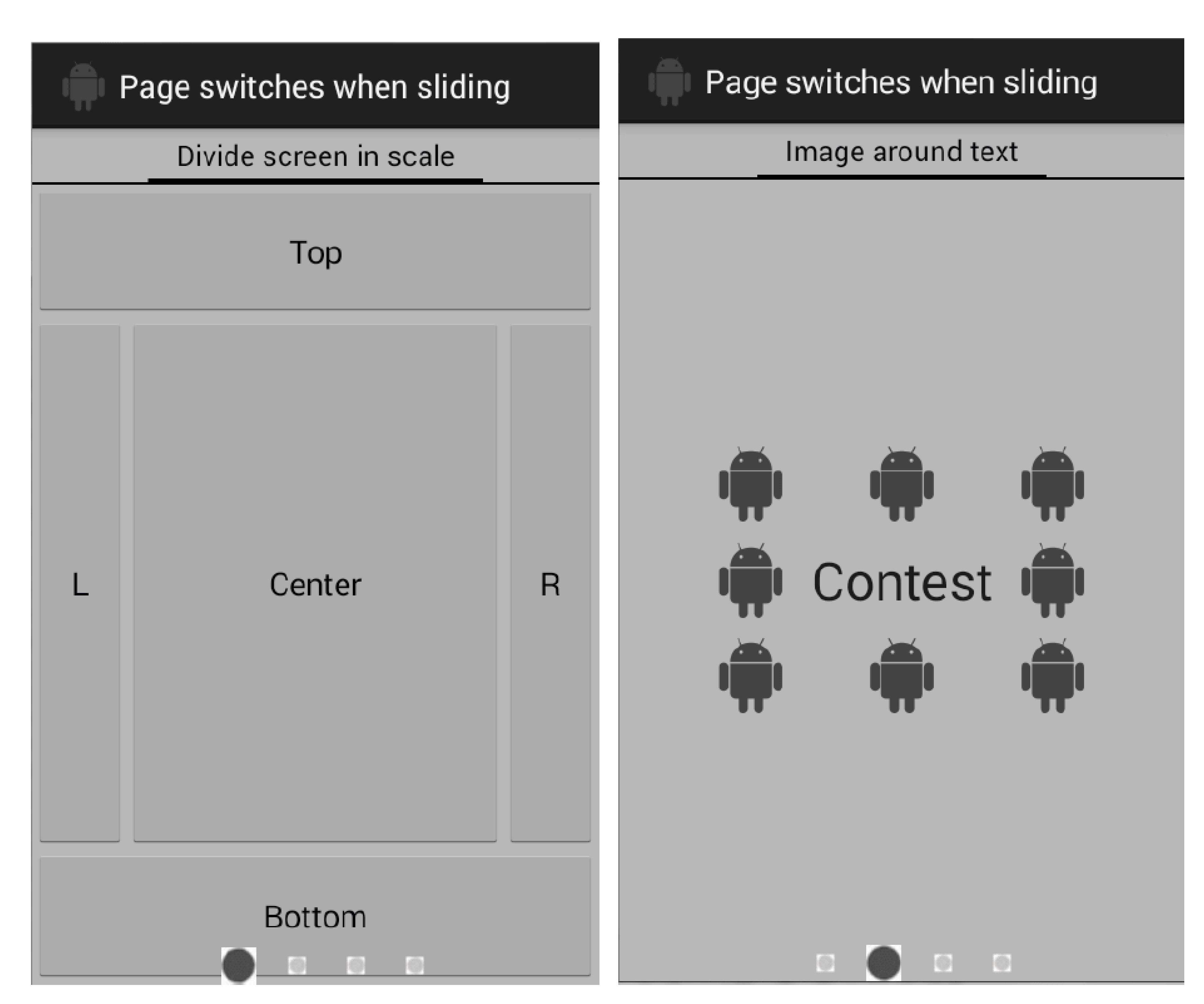


Figure 7-1 the figure of the running results at different pages

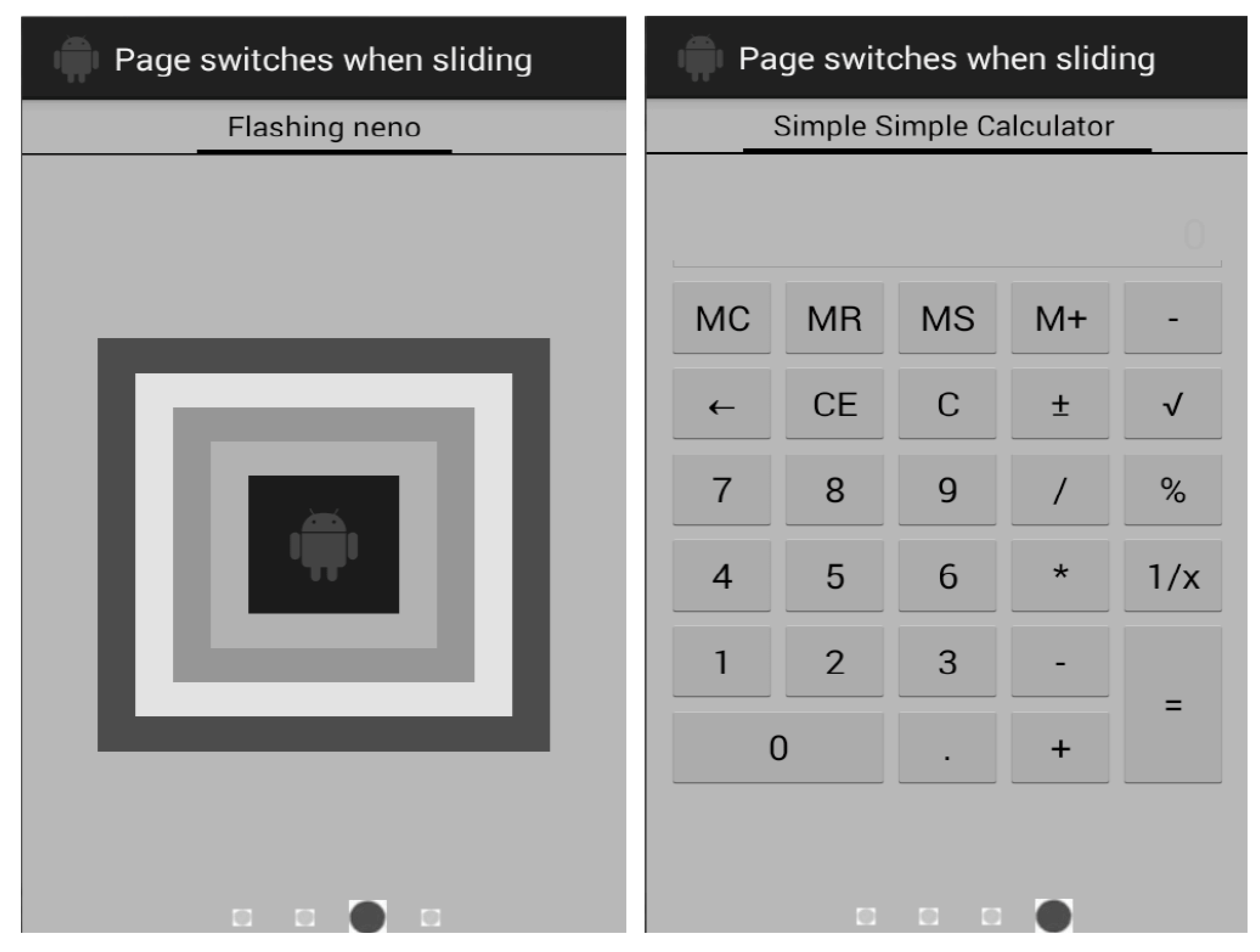


Figure 7-1(continued)

7.2 Key Code

Layout file: 07\PageSwitcher\res\layout\activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc" >
    <android.support.v4.view.ViewPager
        android:id="@+id/mViewPager"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <android.support.v4.view.PagerTabStrip
            android:id="@+id/mTab"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="top" />
    </android.support.v4.view.ViewPager>
    <LinearLayout
        android:id="@+id/mImgs"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:orientation="horizontal" >
    </LinearLayout>
</RelativeLayout>
```

Proportionally split screen page layout file: 07\PageSwitcher\res\layout\linearlayout.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
     <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="@string/up"/>
     <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="4"
        android:orientation="horizontal">
        <Button
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="@string/left"/>
        <Button
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="4"
            android:text="@string/center"/>
        <Button
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="@string/right"/>
     </LinearLayout>
     <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="@string/down"/>
```

```
</LinearLayout>
```

Graphics around text page layout file: 07\PageSwitcher\res\layout\relativelayout.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:padding="10dp"
        android:text="@string/text"
        android:textColor="#0000ff"
        android:textSize="30sp" />
    <ImageView
        android:id="@+id/left"
        style="@style/myImgStyle"
        android:layout_centerVertical="true"
        android:layout_toLeftOf="@id/center"
        android:contentDescription="@string/imageInfo" />
    <ImageView
        android:id="@+id/right"
        style="@style/myImgStyle"
        android:layout_centerVertical="true"
        android:layout_toRightOf="@id/center"
        android:contentDescription="@string/imageInfo" />
    <ImageView
        android:id="@+id/up"
        style="@style/myImgStyle"
        android:layout_above="@id/center"
        android:layout_centerHorizontal="true"
        android:contentDescription="@string/imageInfo" />
    <ImageView
        android:id="@+id/down"
        style="@style/myImgStyle"
        android:layout_below="@id/center"
        android:layout_centerHorizontal="true"
        android:contentDescription="@string/imageInfo" />
    <ImageView
        style="@style/myImgStyle"
        android:layout_alignTop="@id/up"
        android:layout_alignLeft="@id/left"
```

```
        android:contentDescription="@string/imageInfo" />
    <ImageView
        style="@style/myImgStyle"
        android:layout_alignTop="@id/up"
        android:layout_alignLeft="@id/right"
        android:contentDescription="@string/imageInfo" />
    <ImageView
        style="@style/myImgStyle"
        android:layout_alignBottom="@id/down"
        android:layout_alignLeft="@id/left"
        android:contentDescription="@string/imageInfo" />
    <ImageView
        style="@style/myImgStyle"
        android:layout_alignBottom="@id/down"
        android:layout_alignLeft="@id/right"
        android:contentDescription="@string/imageInfo" />
</RelativeLayout>
```

The neon page layout file: 07\PageSwitcher\res\layout\framelayout.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
     <TextView
         android:id="@+id/text01"
         android:layout_width="240dp"
         android:layout_height="240dp"
         android:background="#ff0000"
         android:layout_gravity="center"/>
     <TextView
         android:id="@+id/text02"
         android:layout_width="200dp"
         android:layout_height="200dp"
         android:background="#ffff00"
         android:layout_gravity="center"/>
     <TextView
         android:id="@+id/text03"
         android:layout_width="160dp"
         android:layout_height="160dp"
         android:background="#00ff00"
         android:layout_gravity="center"/>
     <TextView
         android:id="@+id/text04"
         android:layout_width="120dp"
```

```
            android:layout_height="120dp"
            android:background="#00ffff"
            android:layout_gravity="center"/>
        <TextView
            android:id="@+id/text05"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:background="#0000ff"
            android:layout_gravity="center"/>
        <ImageView
            android:src="@drawable/ic_launcher"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:contentDescription="@string/imageInfo"/>
</FrameLayout>
```

Calculator page layout file: 07\PageSwitcher\res\layout\gridlayout.xml

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:padding="10dp"
    android:columnCount="5">
    <EditText
        android:layout_height="60dp"
        android:layout_columnSpan="5"
        android:enabled="false"
        android:gravity="right|bottom"
        android:inputType="numberDecimal"
        android:textSize="24sp"
        android:text="@string/zero"
        android:layout_gravity="fill_horizontal"/>
    <Button
        style="@style/myStyle"
        android:text="@string/mc"/>
    <Button
        style="@style/myStyle"
        android:text="@string/mr"/>
    <Button
        style="@style/myStyle"
        android:text="@string/ms"/>
    <Button
```

```
        style="@style/myStyle"
        android:text="@string/mplus"/>
    <Button
        style="@style/myStyle"
        android:text="@string/minus"/>
    <Button
        style="@style/myStyle"
        android:text="@string/arrow"/>
    <Button
        style="@style/myStyle"
        android:text="@string/ce"/>
    <Button
        style="@style/myStyle"
        android:text="@string/c"/>
    <Button
        style="@style/myStyle"
        android:text="@string/plusminus"/>
    <Button
        style="@style/myStyle"
        android:text="@string/correct"/>
    <Button
        style="@style/myStyle"
        android:text="@string/seven"/>
    <Button
        style="@style/myStyle"
        android:text="@string/eight"/>
    <Button
        style="@style/myStyle"
        android:text="@string/nine"/>
    <Button
        style="@style/myStyle"
        android:text="@string/div"/>
    <Button
        style="@style/myStyle"
        android:text="@string/mode"/>
    <Button
        style="@style/myStyle"
        android:text="@string/four"/>
    <Button
        style="@style/myStyle"
        android:text="@string/five"/>
    <Button
        style="@style/myStyle"
        android:text="@string/six"/>
```

```
    <Button
        style="@style/myStyle"
        android:text="@string/mul"/>
    <Button
        style="@style/myStyle"
        android:text="@string/daoshu"/>
    <Button
        style="@style/myStyle"
        android:text="@string/one"/>
    <Button
        style="@style/myStyle"
        android:text="@string/two"/>
    <Button
        style="@style/myStyle"
        android:text="@string/three"/>
    <Button
        style="@style/myStyle"
        android:text="@string/minus"/>
    <Button
        style="@style/myStyle"
        android:text="@string/equal"
        android:layout_rowSpan="2"
        android:layout_height="100dp"/>
    <Button
        style="@style/myStyle"
        android:text="@string/zero"
        android:layout_columnSpan="2"
        android:layout_width="120dp"/>
    <Button
        style="@style/myStyle"
        android:text="@string/dot"/>
    <Button
        style="@style/myStyle"
        android:text="@string/plus"/>
</GridLayout>
```

The constant string file is easy,so not listed here.

Add the following code in the style file :07\PageSwitcher\res\values\styles.xml

```
<style name="myStyle">
    <item name="android:layout_width">60dp</item>
<item name="android:layout_height">50dp</item>
<item name="android:textSize">20sp</item>
<item name="android:gravity">center</item>
```

```
</style>
<style name="myImgStyle">
    <item name="android:layout_width">wrap_content</item>
<item name="android:layout_height">wrap_content</item>
<item name="android:src">@drawable/ic_launcher</item>
</style>
```

The main program code:

07\PageSwitcher\src\iet\jxufe\cn\android\pageswitcher\MainActivity.java

```
public class MainActivity extends Activity {
    private ViewPager mViewPager;//the widget to save several pages
    private PagerTabStrip mTab; //page TAB
    private LinearLayout mImgs;//LinearLayout to store the bottom images
    private int[] layouts = new int[] { R.layout.linearlayout,
                   R.layout.relativelayout, R.layout.framelayout,
                   R.layout.gridlayout }; //layouts for pages
    private String[] titles = new String[] { "Divide screen in scale",
                            "Image around text","Flashing neno",
                            "Simple Simple Calculator "};
    private ImageView[] mImgViews = new ImageView[layouts.length];
    private List<View> views = new ArrayList<View>();
                                   //the collection to store views
    private List<String> pagerTitles = new ArrayList<String>();
                                  //the collection to store titles
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //set the page to full screen
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                   WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_main);
        //get view by the id of view
        mImgs = (LinearLayout) findViewById(R.id.mImgs);
        mViewPager = (ViewPager) findViewById(R.id.mViewPager);
        mTab = (PagerTabStrip) findViewById(R.id.mTab);
        //the margin between tabs, by default, you can see multiple tabs
          in a page
        mTab.setTextSpacing(200);
        init();//to perform initialization
        mViewPager.setAdapter(new MyPagerAdapter());//add adapter for
                                               ViewPager controls
        initImg();//initialize the display image
        //add a page change event listeners for ViewPager controls
```

```
            mViewPager.setOnPageChangeListener(new
                                        MyPageChangeListener());
    }
    //custom page transformation event listeners
    private class MyPageChangeListener implements OnPageChangeListener {
        public void onPageScrollStateChanged(int arg0) {
        }
        public void onPageScrolled(int arg0, float arg1, int arg2) {
        }
        //invoke when the page of show has changed
        public void onPageSelected(int selected) {
            //reset the images in the bottom
            resetImg();
            //set the color of current page to red
                mImgViews[selected].setImageResource(R.drawable.choosed);
        }
    }
    //custom class which is used to encapsulate the pages will switch
    private class MyPagerAdapter extends PagerAdapter {
        //method used to get the number of pages
        public int getCount() {
            return views.size();
        }
        public boolean isViewFromObject(View arg0, Object arg1) {
            return arg0 == arg1;
        }
        //method used to get the title of the page
        public CharSequence getPageTitle(int position) {
            return pagerTitles.get(position);
        }
        //method used to initialize the specified page
        public Object instantiateItem(ViewGroup container, int
                                            position) {
            ((ViewPager) container).addView(views.get(position));
            return views.get(position);
        }
        //method used to destroy the specified page
        public void destroyItem(ViewGroup container, int position,
                                    Object object) {
            ((ViewPager) container).removeView(views.get(position));
        }
    }
    //method used to initialize the page, and add it to the collection
    public void init() {
```

```
        for (int i = 0; i < layouts.length; i++) {
            View view = getLayoutInflater().inflate(layouts[i], null);
            views.add(view);
            pagerTitles.add(titles[i]);
        }
    }
    //method used to initialize the bottom images, and add them to horizontal LinearLayout
    public void initImg() {
        for (int i = 0; i < mImgViews.length; i++) {
            mImgViews[i] = new ImageView(MainActivity.this);
            if (i == 0) {//the first picture is selected by default
                mImgViews[i].setImageResource(R.drawable.choosed);
            } else {
                mImgViews[i].setImageResource(R.drawable.unchoosed);
            }
            mImgViews[i].setPadding(20, 0, 0, 0);
            mImgViews[i].setId(i);
            mImgViews[i].setOnClickListener(mOnClickListener);
            mImgs.addView(mImgViews[i]);
        }
    }
    private OnClickListener mOnClickListener = new OnClickListener(){
        public void onClick(View v) {
            resetImg();//reset the image, set the color of images to yellow
            //set the color of the image which clicked to red
            ((ImageView) v).setImageResource(R.drawable.choosed);
            //change the display page which according to clicked image
            mViewPager.setCurrentItem(v.getId());
        }
    };
    //method used to set the state of all the ImageView to unselected
    public void resetImg() {
        for (int i = 0; i < mImgViews.length; i++) {
            mImgViews[i].setImageResource(R.drawable.unchoosed);
        }
    }
}
```

7.3 Code Analysis

7.3.1 Interface analysis

It is clear that there are mainly two widgets in the interface, the one is ViewPager widget and the other one is LinearLayout widget, whole adopt RelativeLayout. The ViewPager filled the screen used to dynamically display the page. When scrolling the pages, the ViewPager widget will remove the previous page, and reload new page. The LinearLayout widget is located center horizontally at the bottom of screen, which is used to display the image widgets. But due to the image widget is dynamically changed, we dynamically specified this part of information in the code; The LinearLayout is empty temporarily, we just specified its direction and alignment. The code is listed in activity_main.xml. Since ViewPager class is in the compatible package provided by the Android system and it is similar to the third party jar package, so we can not directly use it. We need to write the full package name and class name, otherwise the system can not recognize.

This case used some previous examples and lists the codes without detail information. Please check the detail information of the previous examples if you have any questions.

7.3.2 ViewPager

Users can easily switch the pages through the ViewPager, just swipe your finger left or right on the screen. The ViewPager is a container in essence, you can add and remove widgets in the container. The widgets can be some simple widgets such as buttons, TextView widgets, etc. It also can be complex widgets, such as the custom View objects, containers, etc. In fact, a page can also viewed as a complex View object, usually we put the design of the page in the layout file, then call the getLayoutInflater (). Inflate () method in activity used to convert the layout file into a View object, then we can add the View objects into the ViewPager container.

In Android, the ViewPager object itself cannot directly connect with the page; it uses PagerAdapter to connected with the page. The PagerAdapter class is an abstract class, it cannot be instantiated directly, and the Android system provides a series of implement class, such as FragmentPagerAdapter, FragmentStatePagerAdapter. What's more, we can define the implement class of the PagerAdapter class. Here we introduce the common method that defines a subclass of a PagerAdapter class. The subclass at least needs to implement, the following four methods:

- **instantiateItem(ViewGroup container, int position)**: This method is used to initialize the specified page;
- **destroyItem(ViewGroup container, int position, Object object)**: The method used to destroy the specified page;

- **getCount()**: This method returns the number of pages that can be switched;
- **isViewFromObject(View arg0, Object arg1)**: This method is used to determine whether View objects and Object for the same Object.

In addition you can also override the getPageTitle (int position) method to obtain the title of specified page , etc. The codes define the subclass of PagerAdapter from 42th-63th line.

After creating the PagerAdapter objects, you can combined the ViewPager object with the PagerAdapter object together by calling the setAdapter () method of ViewPager object. After this process, the page is associated with the ViewPager. Now we realize the effect that switching the pages. Until this, the switching pages have not established relationship with the ImageView widget.

We want the ImageView widget changed after switching the pages. Here we add event listeners to the ViewPager, which means we call the setOnPageChangeListener () method of the ViewPager. This method need to pass specific event listener object which means we need to implement the OnPageChangeListener. There are three methods in the event listener, one used to monitor the page scroll, one used to monitor the status changed when scrolling, one used to monitor the selected page. In this case, we just handle the selected page.

When we handle the event, firstly, we should make all the ImageView widgets change to the unselected icons. Secondly, we set the currently displayed ImageView widget to the selected icon. Specific codes are listed from 28th to 40th line.

In this case we can scroll screen to switch the pages, we can also click the ImageView widget jump to the corresponding page which located at the bottom of screen. The realization process: firstly, providing the event listener to the ImageView widgets when initialize them; Secondly, changing the current page to the corresponding page. The codes initialize ImageView widgets are from 73th-86th line; the codes click event handler form 89th-97th line.

7.4 Expansion of Knowledge

7.4.1 Event handler based on listening

In this case mainly introduced two kinds of event handler, they are click event handler of ImageView widget , and page change event handler of ViewPager object. They all use event listener to monitor the event, then process the method in event and listener will handle it. So what is a event listener? How is the process of event handler based on listening?

Android's event handler model is based on listening, and it seems like the process of Java AWT and Swing. Just the event listeners and event handler methods are different. In the event handler model based on listening, it mainly has three kinds of objects:

EventSource: That is the source of the event, usually represented as the widget like Buttons, ImagesView, ListsView, etc;

Event: The detailed description of users'operation, and the event encapsulate the information of the operation. Usually use Event object to get the related event information from the EventSource. For example, key events which the key has been pressed, touch event where the touch event occurred,etc;

EventListener: Monitor the users' operation on the Event Source, and give the corresponding response to users' various operation. An event listener contains some event handlers, actually one event handler is defined as one event-handling methods.

In the process of event-handling based on listening, what it is like when the three objects collperate together? In fact, event-handling based on listening is a kind of delegation event model. The EventSource entrust the whole event handler to a specific object EventListener, and the EventSource is a normal widget; the system generates a Event object automatically and sends a message to the EventListener, the corresponding EventHandler in the EventListener will handle the Event When a specific Event happened on the EventSource. An event handling model showed below in Figure 7-2.

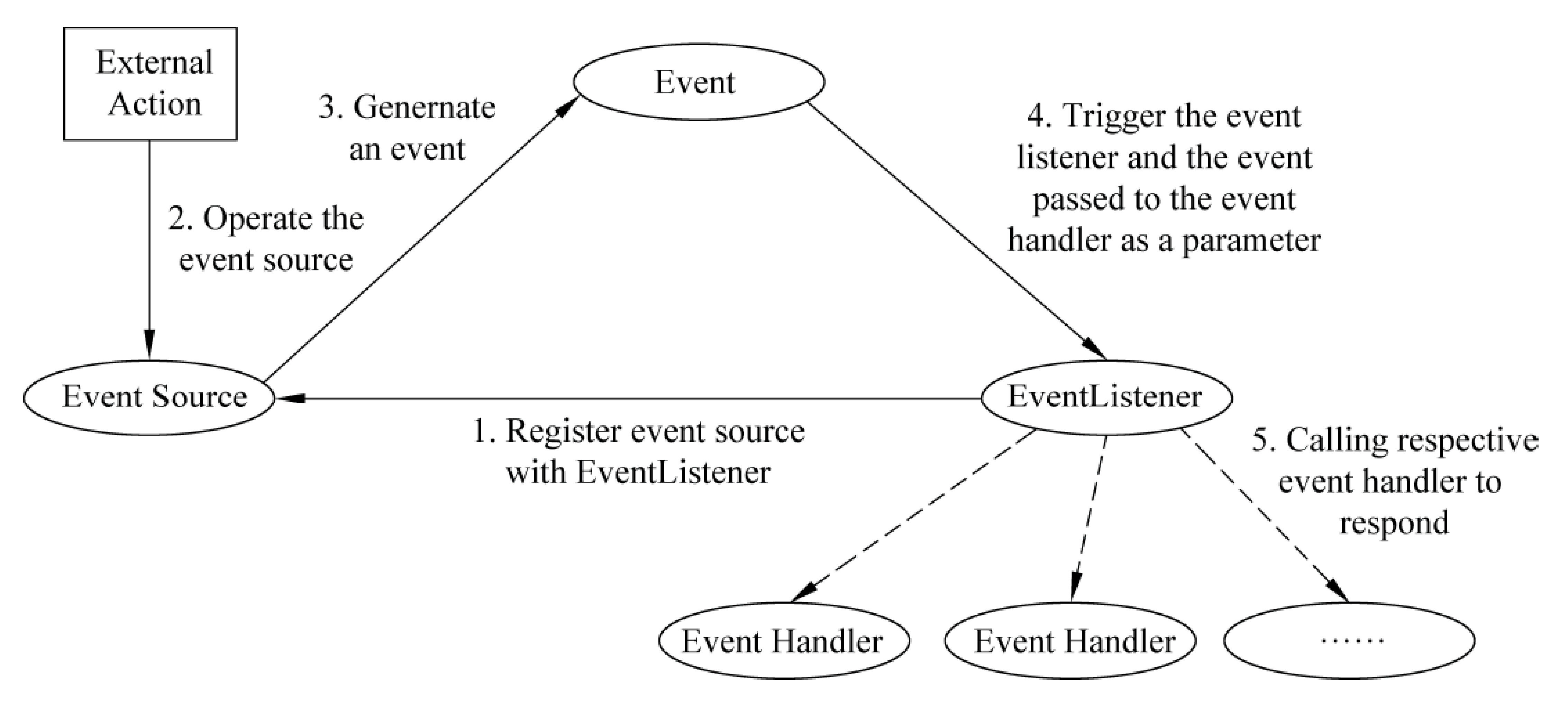


Figure 7-2　the event handling model based on listening

The system generates an Event object automatically and passes the object in parametric form to the EventListener which registed on the Event Source, the corresponding EventHandler will handle the Event when users perform operation on the widget. This process seems like that everyone's ability is limited when we meet something cannot handle, and then we entrust an intermediary agent to deal with it. We just need to describe the things and the requirement clearly, so that the intermediary agent can be better to handle these things, like the intermediary agent will send an intermediary to help handle the things.

Ourselves are the event source, the thing is the Event, the intermediary agent is the EventListener, intermediary helping us handle things is the EventHandler.

The programming steps of the event-handling model based on listening as followed:

(1) Get the widget (EventSource), the object which is being monitored;

(2) Implement an EventListener class, the EventListener class is a special Java class,

which must implement an XxxListerner interface and implement all the methods in the interface, each method is used to handle one event;

(3) Combined the EventSource with the EventListener together that register a EventListener to a widget(EventSource) through the setXXXListener method of the EventSource.

In the above steps, we can easily get the EventSource according to the findViewById () method, usually is the interface widget; The Android system has already defined the setXXXListener of EventListener, we just need to pass a specific EventListener. The Implement of the EventListener is the key of EventHandler based on listening. The so-called EventListeners, actually is an instance which has implement specific Java interface class. Thus, the following several forms are used to realize the EventListener in the program.

- **Inner from of class**: The EventListener class is defined as an inner class of the current class;
- **External form of class**: The Event Listener class is defined as an external class;
- **The class itself as an Event Listener class**: The Activity itself realize the listener interface and implement the event-handling methods;
- **An anonymous inner class**: Using an anonymous inner class to create an EventListener object.

7.4.2 Page full screen

By default, the phone interface contains three parts, they are the status bar, the title bar, and the content. In order to get full value from the screen and make the screen looks more beautiful, you can use the full screen mode when developing an android application, especially the development of the game title bar and the status bar will not be displayed.

a. There are two methods to avoid the title bar in the Android.

The first, setting in the Java codes, adding the following code before the setContentView () method:

```
requestWindowFeature(Window.FEATURE_NO_TITLE);
```

The second, adding android: theme =**"@android:style/Theme.NoTitleBar"** to the corresponding activity within the <manifest> block.You can also set other values contain NoTitleBar.

b. There are also two methods to avoid the status bar in the Android.

The first, setting in the Java codes, adding the following code before the setContentView () method:

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

The second, adding android:theme=“@android: style/ Theme.NoTitleBar” to the corresponding activity within the <manifest> block. You can also set other values contain Fullscreen.

7.5 Thinking and Exercises

(1) In this case, we just remove the status bar, and try to remove the title bar and status bar at the same time.

(2) Event-handling model is based on listening, what is the mainly kinds of object?

(3) Describe of the process of Event-handling is based on listening ?

(4) The implement method of EventListener ________.________.________and_________.

(5) Which kind of design patterns as showed below is used in the thinking of event-handling based on listening? (   )

A. observer pattern   B. proxy pattern   C. strategy pattern   D. decorator pattern

Chapter 8 Images Switch Automatically

8.1 Case Overview

This case mainly introduced the effect of pictures switching at a regular time every eight seconds that the current picture will changed to another picture at the same time, you can see the process of switch pictures. The circles are under the picture connected with the corresponding pictures. In which the red circle shows the current picture, and the status of circles will change at the same time when the picture is changed; you can also directly click the circle jump to the corresponding picture. What’s more, this case also supports scrolling gesture operation switch pictures according to the direction and speed of gestures. Our application running looks like Figure 8-1.



Figure 8-1 the figure of the running results at different times

8.2 Key Code

Layout file: 08\ImageScan\res\layout\activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc"
    android:orientation="vertical">
     <iet.jxufe.cn.android.imagescan.MyImageTopView
        android:layout_width="match_parent"
        android:layout_height="240dp"
        android:id="@+id/mTopView">
     </iet.jxufe.cn.android.imagescan.MyImageTopView>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal"
        android:id="@+id/mBottomView"
        android:background="#ffffff">
    </LinearLayout>
</LinearLayout>
```

Program code: 08\ImageScan\src\iet\jxufe\cn\android\imagescan\MainActivity.java

```
public class MainActivity extends Activity {
    private MyImageTopView mTopView;//custom view used to display the
                                //picture container
    private LinearLayout mBottomView;//LinearLayout used to show the
                                //circle in the bottom
    private int[] imgIds = new int[] { R.drawable.pic1, R.drawable.pic2,
            R.drawable.pic3, R.drawable.pic4, R.drawable.pic5,
            R.drawable.pic6,
            R.drawable.pic7 };//the ids of images
    public ImageView[] imgViews = new ImageView[imgIds.length];
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mBottomView = (LinearLayout) findViewById(R.id.mBottomView);
        mTopView = (MyImageTopView) findViewById(R.id.mTopView);
        initBottom();//initialize circles at the bottom, by default, the
                    first one is selected
        mTopView.initImages(imgIds);//initialize the images
    }
    //initialize the circles at the bottom and add click event handler
    //for them
    public void initBottom() {
        for (int i = 0; i < imgViews.length; i++) {
```

```
                imgViews[i] = new ImageView(this);
                if (i == 0) {
                    imgViews[i].setImageResource(R.drawable.choosed);
                } else {
                    imgViews[i].setImageResource(R.drawable.unchoosed);
                }
                imgViews[i].setPadding(15, 0, 0, 0);//setting the margin
                                                      //between circles
                imgViews[i].setId(i);//set the id for the circle
                imgViews[i].setOnClickListener(new OnClickListener() {
                    public void onClick(View v) {
                        resetImg();//set all circles unselected
                        ((ImageView) v).setImageResource(R.drawable.choosed);
                         //set current circle selected
                        mTopView.scrollToImage(v.getId());
                    //switch image to the one which match current circle
                    }
                });
                mBottomView.addView(imgViews[i]);//add all the circles to
                                                   //the linearlayout
            }
        }
        //method used to set all the circles unselected
        public void resetImg() {
            for (int i = 0; i < imgViews.length; i++) {
                imgViews[i].setImageResource(R.drawable.unchoosed);
            }
        }
}
```

Program code: 08\ImageScan\src\iet\jxufe\cn\android\imagescan\MyImageTopView.java

```
public class MyImageTopView extends ViewGroup {
    private GestureDetector gesDetector;
    private Scroller scroller;
    private int currentImageIndex=0;//record the index of current image
    private boolean fling = false;//judge mark, prevent onTouch event
                                   //handling UP events repeat
    private Handler mHandler;//use to send, receive and process the
                              //message
    private Context context;
    public MyImageTopView(Context context, AttributeSet attributeSet){
        super(context, attributeSet);
        this.context=context;
        init();
```

```
        this.setOnTouchListener(new MyOnTouchListener());
                                        //add a touch event listener
    }
    public void init() {
        scroller = new Scroller(context);//Create a scroll bar
        mHandler = new Handler(){//create Handler object and override
                                    //its message processing method
            public void handleMessage(Message msg) {
                if(msg.what==0x11){
                    //switch to the specified image
                    scrollToImage((currentImageIndex + 1) %
                                    getChildCount());
                }
            }
        };
        gesDetector = new GestureDetector(context, new
                                    OnGestureListener() {
            //Notified when a tap occurs with the up MotionEvent
            public boolean onSingleTapUp(MotionEvent e) {
                return false;
            }
            //The user has performed a down MotionEvent and not performed
            //a move or up yet
            public void onShowPress(MotionEvent e) {
            }
            public boolean onScroll(MotionEvent e1, MotionEvent e2,
                    float distanceX, float distanceY) { //slide your
                                    //fingers on the touch screen
                //If the sliding range between the first page and last
                //page, distanceX>0 represent slide to the right,
                //distanceX < 0 represent scroll to left
                //If out of the range, don't do any action
                If ((distanceX > 0 && getScrollX() < getWidth()
                        * (getChildCount() - 1))
                        || (distanceX < 0 && getScrollX() > 0)) {
                    scrollBy((int) distanceX, 0);
                    //the distance to Scroll, only considering
                    //horizontal scrolling, Vertical scrolling is 0
                }
                Return true;
            }
            public void onLongPress(MotionEvent e) {
        //Notified when a long press occurs with the initial on down
            }
            public boolean onFling(MotionEvent e1, MotionEvent e2,
```

```
                        float velocityX, float velocityY) {
            //Fingers on the touch screen mobile quickly and loosen
                //Determine whether minimum arrive sliding speed, save
                //absolute value
                if (Math.abs(velocityX) > ViewConfiguration.get(context)
                        .getScaledMinimumFlingVelocity()) {
                        //if faster than the minimum speed
                    if (velocityX > 0 && currentImageIndex >= 0) {
                        fling = true;//velocityX>0 means sliding to the
                                //left
                        scrollToImage((currentImageIndex - 1 +
                                getChildCount())
                                % getChildCount());
                    } else if (velocityX < 0
                            && currentImageIndex <= getChildCount() - 1){
                        fling = true;//velocityX<0 means sliding to the
                                //right
                        scrollToImage((currentImageIndex + 1)
                                % getChildCount());
                    }
                }
                return true;
            }
            public boolean onDown(MotionEvent e) {
            //Notified when a tap occurs with the down MotionEvent
                return false;
            }
        });
        Timer timer=new Timer();//create a timer object
        timer.schedule(new TimerTask() {
         //cycle operation, every eight seconds sends a message
            public void run() {
                mHandler.sendEmptyMessage(0x11);
            }
        }, 0,8000);
    }

    public void scrollToImage(int targetIndex){//jump to the target image
        if (targetIndex != currentImageIndex && getFocusedChild() != null
                && getFocusedChild() == getChildAt(currentImageIndex)) {
            getFocusedChild().clearFocus(); //the current image clear focus
        }
        final int delta =targetIndex * getWidth() - getScrollX();
         //The distance to sliding
        int time =Math.abs(delta) * 5;
```

```
        //time represent sliding time, is five times of the distance to slide
        scroller.startScroll(getScrollX(), 0, delta, 0, time);
        //start to slide
        invalidate();//refresh the page
        currentImageIndex = targetIndex; //change the index of current
                                          //image
        ((MainActivity)context).resetImg();
        //change the state of the circle
    MainActivity) context).imgViews [currentImageIndex].
    setImageResource(R.drawable.choosed);
    }
    public void computeScroll() {//override the superclass
                    //method,record the new position of the scroll bar
        super.computeScroll();
        if (scroller.computeScrollOffset()) {
            scrollTo(scroller.getCurrX(), 0);
            postInvalidate();
        }
    }
    private class MyOnTouchListener implements OnTouchListener{
     //touch event listener
        public Boolean onTouch(View v, MotionEvent event) {
            gesDetector.onTouchEvent(event);
            //will touch events to GestureDetector to deal with
            if (event.getAction() == MotionEvent.ACTION_UP) {
                if (!fling) {//when the user stops dragging
                    snapToDestination();
                }
                fling = false;
            }
            return true;
        }
    }
    //The method is inherited from the ViewGroup, used to specify the
    //child view how to put in container, when the child view size change
    //will the callback the method
    protected void onLayout(boolean changed, int left, int top, int right,
            int bottom) {
        //set layout, the child view order transverse alignment
        for (int i = 0; i < getChildCount(); i++) {
            View child = getChildAt(i);//access to each child view
            child.setVisibility(View.VISIBLE);//set the control is visible
            child.measure(right - left, bottom - top);
            child.layout(i * getWidth(), 0, (i + 1) * getWidth(),
                    getHeight());
```

```
            }
        }
        private void snapToDestination() {
            scrollToImage((getScrollX() + (getWidth() / 2)) / getWidth());
             //rounding, more than half into next picture
        }
        //initialize the display image
        public void initImages(int[] imgIds) {
            int num = imgIds.length;//get the length of image collection
            this.removeAllViews();//remove all child views
            for (int i = 0; i < num; i++) {//add imageView one by one
                ImageView imageView = new ImageView(getContext());
                imageView.setImageResource(imgIds[i]);//set the image for
                                                       //each imageView
                this.addView(imageView);//add images to the custom widget
            }
        }
}
```

8.3 Code Analysis

8.3.1 Interface analysis

It is clear there are mainly two widgets in the interface, they are one custom MyImageTopView widget and one LinearLayout widget, they are both overall adopt vertical LinearLayout. In which the MyImageTopView widget is mainly used to display the pictures available for switching at the top of the screen. It is a custom widget, and inherits from the ViewGroup. It can store several widgets; the widget is placed from left to right and next to each other in a container. The LinearLayout widget located center horizontally under the MyImageTopView widget, it is used to show the circles. Due to the circles are dynamically changed, so that some information need to be dynamically specified in the codes. The LinearLayout widget just specified the direction of the LinearLayout and the alignment of the content, it is temporarily empty. The codes are listed in activity_main.xml.

Since the MyImageTopView widget is a user-defined widget, therefore, you need to use the full package name + class name when using this widget in the layout file, otherwise, the system cannot be recognized.

8.3.2 Custom MyImageTopView widget

The MyImageTopView widget inherits from the ViewGroup class, you need to implement its abstract method onLayout (), this method is used to specific how to place the widgets in the

container. When the widget size is changed, the widgets will callback this method. This method specified the widgets in container placed form left to right, which means they can be placed in horizontal direction according to the numbers of widgets. The widgets placed form top to bottom in vertical direction. The specific codes are listed in the file MyImageTopView.java from 109th-118th line.

The MyImageTopView widget is mainly used in switching pictures. First, you need to initialize some pictures for it. Here we write a method that initImages(int[]imgIds) and create corresponding ImageView widgets according to the numbers of pictures in the array, then add this method into the container, the code is listed in the file MyImageTopView.java from 123rd-131st line.

We use the timer regularly to switch the pictures. The timer perform an operation, which means sending a message through the Handler object in every 8000 milliseconds, then the Handler object received the message, and call the method which performs the slide to next picture. The timer codes: from 67th-72th line, the handler message codes: from 16th-23rd line.

In addition, this case also supports gestures, so you need to add TouchEvent handler for the MyImageTopView widget; then pass the TouchEvent to the Gesture Detector in the MyOnTouchListener. The TouchEvent handler codes: form 96th-107th line, the Gesture Detector codes: from 24th-66th line. Gestures mainly identify the sliding distance, direction.etc. Then according to this system, determining which picture to show. In addition we also need to override the computeScroll () method of the superclass so that it can slide to the specified position in time.

The specific method of the slide operation: the scrollToImage (int targetIndex), this method need to pass a parameter the numbers of the target pictures. The process of internal execution: First, work out the sliding distance, then set the sliding time, usually the sliding distance has a relationship with the sliding time. Here we set the sliding time to be 5times of the sliding distance, then call the sliding startScroll() method of the scroll bar. This method need to pass five parameters, the x and y coordinates of the start point, the x and y coordinates of the sliding distance, the sliding time. After sliding to the target page, the circles under the pictures also need to be changed. However the circles are not parts of the MyImageTopView widget, here refer to the interaction between the MyImageTopView widget and the MainActivity. We need to pass a context parameter, actually it is the MainActivity. We just caste the context parameter, then call the related members and the methods of MainActivity. The code is listed from 75th-88th line.

8.4 Expansion of Knowledge

8.4.1 Custom widget

In Android, all the widgets are inherited from the View class, the View class provides some common properties, similar to the Java Object class is the superclass of all classes. You can suppose the View object like a rectangular area on the screen; Different widgets inherit the View class and generate more powerful widgets by overriding some methods of the View class or adding some methods. Developers can create their own style widgets by inheriting the View class. The steps of develop a custom widget as followed:

(1) Define the class name of your widget and let the class inherit the View class or a subclass of an existing View class. In this case, we want to create a container widget store other widgets, so let the class inherits the ViewGroup class which is a subclass of the View class.

(2) To add a constructor for the custom widget, then selectively override the method of superclass. When you create a custom widget, you need to provide an explicitly constructor. The constructor is the basic way to create a custom widget. No matter you created the widget in java codes or in layout files, the constructor will be executed. The superclass itself does not have the no-argument constructor; the default no-argument constructor in the subclass will call the no-argument constructor in the superclass and which will make an error. The constructor method must be provided, and you need to override some methods in the superclass according to the business requirement. Such as the onDraw () method used for drawing interface. In this case we have override the onLayout () method and the computeScroll () method of superclass, these methods called by system automatically.

(3) After you have defined the widget, you can use the widget like using a system widget you can create the widget by the View class in the codes, and also can create the widget by the labels in the layout file. In mind in the layout file, the label name of the widget also contains the full package name and the full class name, not only the original class name. And when you defined the widget, the constructor needs to pass the parameter which type is AttributeSet.

8.4.2 Gesture Detection

The gesture means the act of touching screen with users'finger or the touching pen. For example, touching and sliding from left to right on the screen. Android system provides gesture detection and the corresponding listener. The Gesture detection class in android is the GestureDetector class. When you create an object of this class, you need to pass at least two parameters, one is the current context object, and the other is the gesture listener OnGestureListener. There are several methods in the OnGestureListener used for listening different operations of user. Mainly methods as followed:

- **boolean onDown(MotionEvent e)**: Finger has just come into contact with the moment

of touch screen, triggering the method;

- **void onShowPress(MotionEvent e)**: Fingers on the touch screen, triggering the method, It's time you press the onset, long before pressing;
- **boolean onSingleTapUp(MotionEvent e)**: Fingers away from the moment of touch screen, triggering the method;
- **boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY)**: This method is triggered when the finger slides on the touch screen, this method four parameters denote that the first down motion event that started the scrolling, the move motion event that triggered the current onScroll, the distance along the X axis that has been scrolled since the last call to onScroll, the distance along the Y axis that has been scrolled since the last call to onScroll;
- **void onLongPress(MotionEvent e)**: Finger on the screen for a while, and not loosen is triggered when using the method;
- **boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)**: Fingers on the touch screen mobile quickly, and loosen the triggered when the method, the method and four parameters denote that the first down motion event that started the fling. The move motion event that triggered the current onFling. The velocity of this fling measured in pixels per second along the x axis. The velocity of this fling measured in pixels per second along the y axis.

Gesture detection in Android only need two steps:

(1) **Create a GestureDetector object, and specified the touch listener;**

(2) **Add the TouchEvent listener for the current page or the specific widget**. In this case, adding a TouchEvent listener for the MyImageTopView widget, then give the TouchEvent to the GestureDetector to handle in the TouchEvent handler.

The system will automatically generate a touch event MotionEvent when users touch the screen. The event is monitored by the OnTouchListener, then call the onTouch () method of the OnTouchListener, and pass the MotionEvent as a parameter to the on Touch() method. However, the onTouch()method call the onTouch() method of GestureDetector inside, also pass the MotionEvent as a parameter to the onTouch() method. Once executes the onTouch method, it will be monitored by the gesture listener, and call the related method to handle.

8.5 Thinking and Exercises

(1) Please describe the process from touching screen to perform operation.

(2) Which method did not cleared in the OnGestureListener interface(　)?

A. onDown()　　B. onUp()　　C. onScroll()　　D. onFling

Chapter 9 Keyword Search Tips

9.1 Case Overview

This case prompts the user to type in. When the user enters several characters, the system will automatically match the specified data source based on the user input and display the useful result in the form of a list. The user can choose a particular item to complete the input according to the needs. If the data source doesn't have the user put in, the content entered by user will be saved into the data source and when someone type it again, the content will be displayed as a reminder. Our application running looks like Figure 9-1.

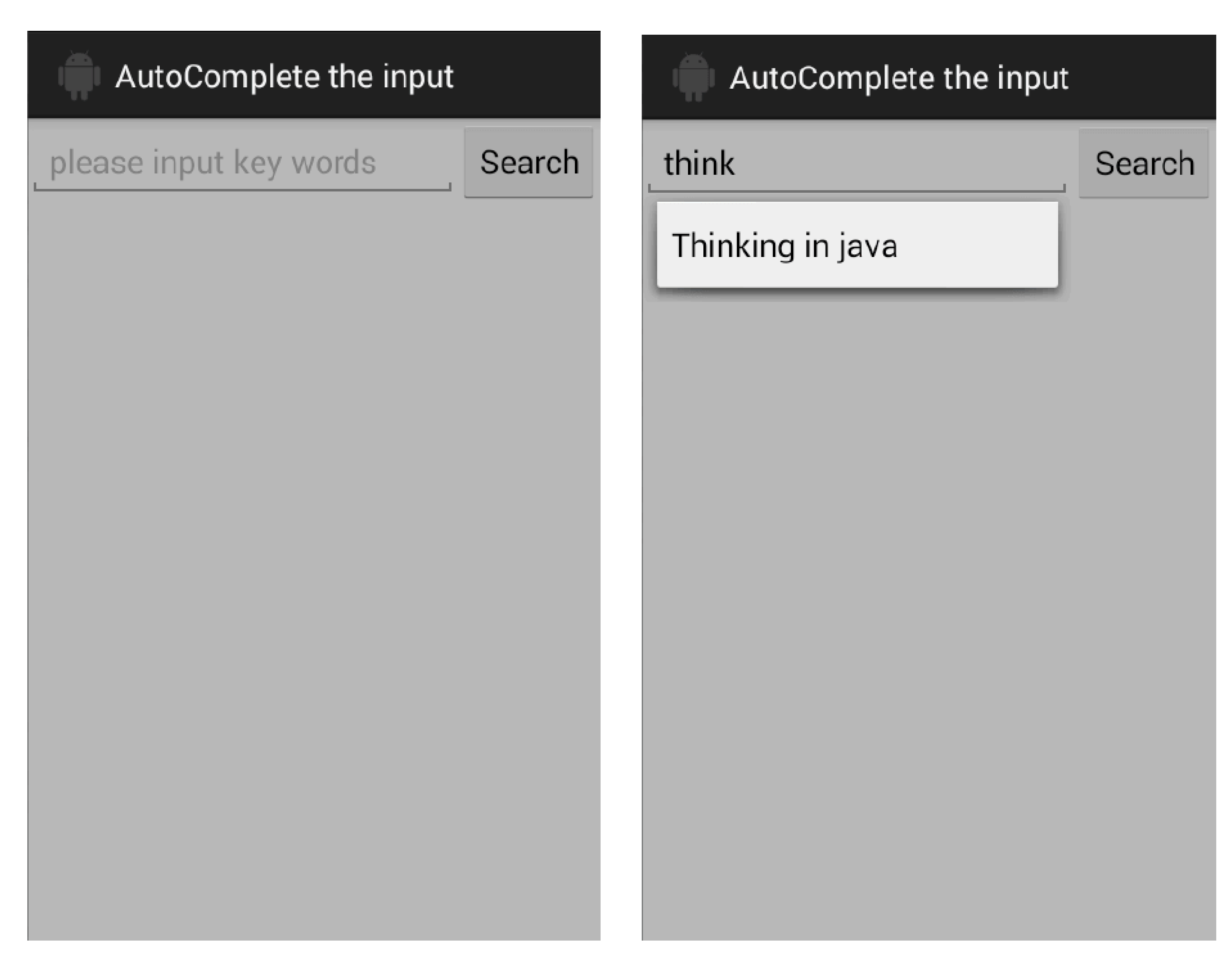


Figure 9-1 the figure of the running results and result of inputing 'think'

9.2 Key Code

Layout file: 09\AutoCompleteTest\res\layout\activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc"
    android:orientation="horizontal">
    <AutoCompleteTextView
        android:id="@+id/mAuto"
        android:layout_width="0dp"
        android:hint="@string/inputHint"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:completionThreshold="1"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="search"
        android:text="@string/search"/>
</LinearLayout>
```

String constant file: 09\AutoCompleteTest\res\values\strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">AutoComplete the input</string>
    <string name="action_settings">Settings</string>
     <string name="search">Search</string>
     <string name="inputHint">please input key words</string>
</resources>
```

Program code: 09\AutoCompleteTest\src\iet\jxufe\cn\android\autocompletetest\MyOpenHelper.java

```
public class MyOpenHelper extends SQLiteOpenHelper {
    public String createTableSQL="create table if not exists word" +
            "(_id integer primary key autoincrement,word text)";
    public MyOpenHelper(Context context, String name, CursorFactory
factory,
            int version) {
        super(context, name, factory, version);
    }
    //invoked when database created, execute the operation of creating
    //table and inserting the original records
    public void onCreate(SQLiteDatabase db) {
db.execSQL(createTableSQL);
        db.execSQL("insert into word(word) values(?)",new String[]
{"Professional Android 4 Application Development"});
        db.execSQL("insert into word(word) values(?)",new String[]
```

```
{"Android from A to D"});
        db.execSQL("insert into word(word) values(?)",new String[]
                {"Android Application Development in 24 Hours"});
        db.execSQL("insert into word(word) values(?)",new String[]
                {"Android Bes t Practices"});
       db.execSQL("insert into word(word) values(?)",new String[]
               {"Thinking in java"});
    }
    //The method will be called back when the version of database is
    //updated
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
                  newVersion) {
        System.out.println("The version changed:" + oldVersion +
                  "------>" + newVersion);
    }
}
```

Program code: AutoCompleteTest\src\iet\jxufe\cn\android\autocompletetest\MainActivity.java

```
public class MainActivity extends Activity {
    private List<String> datas;// collection used to store data
    private AutoCompleteTextView mAuto;
    private MyOpenHelper mHelper;
    private SQLiteDatabase mDB;
    private ArrayAdapter<String> adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mHelper=new MyOpenHelper(this,"word.db", null, 1);
        mDB=mHelper.getReadableDatabase();
        mAuto=(AutoCompleteTextView)findViewById(R.id.mAuto);
        datas=getData();
        adapter=new ArrayAdapter<String>
                (this,android.R.layout.simple_list_item_1,datas);
        mAuto.setAdapter(adapter);
    }
    // Get all the data from database
    public List<String> getData(){
        List<String> contents = new ArrayList<String>();
        Cursor result=mDB.rawQuery("select * from word",null);
        while (result.moveToNext()) {
            contents.add(result.getString(result.getColumnIndex("word")));
        }
```

```
            return contents;
        }
        // the click event handler of search button
        public void search(View view){
            String input=mAuto.getText().toString();
            if(input==null||"".equals(input.trim())){
                AlertDialog.Builder builder=new Builder(this);
                builder.setTitle("Warning");
                builder.setMessage("Please input the key Word!");
                builder.create().show();
            }else{
                // insert the key word into the database if it doesn't exist
                if(!datas.contains(input)){
                    mDB.execSQL("insert into word(word) values(?)",new
                                     String[]{input});
                    datas=getData();
        adapter=new ArrayAdapter<String>
                           (this,android.R.layout.simple_list_item_1,datas);
                    mAuto.setAdapter(adapter);
                }
            }
        }
        @Override
        protected void onDestroy() {
            if(mDB!=null){//close the database when exiting
                mDB.close();
            }
            super.onDestroy();
        }
}
```

9.3 Code Analysis

9.3.1 Smart tips to complete the input

The smart prompt is based on the widget AutoCompleteTextView provided by Android. This widget inherits from EditText. The difference is that according to user input, it can match the specified date source and displayed in the form of a list to the user. The list will refresh from the list when the use enters a character after the list is displayed. Common attributes in AutoCompleteTextView:

- **android:completionThreshold**: Defines the number of characters that the user must type before completion suggestions are displayed in a drop down menu. Must be an

integer value, such as "100". The default value is 2.

- **android:completionHint**: Defines the hint displayed in the drop down menu.
- **android:popupBackground**: Sets the background of the auto-complete drop-down list.
- **android:dropDownVerticalOffset**: Amount of pixels by which the drop down should be offset vertically.
- **android:dropDownHorizontalOffset**: Amount of pixels by which the drop down should be offset horizontally. Default value is left-aligned.

Once we have a widget, we need to assign a data source for it. The data source typically displayed like an array or collection. However, actually data is often stored in a database or file. We use a collection in this case because the data source is dynamically changed. The data collection is retrieved from database so that we need to define a database-related operation class. MyOpenHelper class is a custom database helper class, mainly used to create database, update database and access database. This class inherits from SQLiteOpenHelper class a database helper class provided by Android system, we just need to provide its constructor, and override its onCreate(), onUpdate() method . SQLiteDatabase is a database wrapper class, which provides some common database operations like adding, deleting, searching, changing, etc. and in the next case we will explain it in details.

At this point the data source and the widget are independent, how do they connect with each other? How is the data source displayed? Android provides us an intermediate solution – Adapter. When we create adapter, we need to pass data source, widgets to display each item of the data source and some other parameters so that adapter can specify the display style of data source explicitly. Then the system provides a method called setAdapter() to combine the adapter with widget. Thus, the data source will be able to establish an association with controls and this approach allows the widget and the data sufficiently decoupled, we use the Adapter to display the data.

9.3.2 Intelligent update the data source

After the user enters something and clicks the search Button, the system will check automatically whether the key word is in the data source or not. If not, then add to the data source else without actions. When inserting data into the database, we call a method related to the SQLiteDatabase. Either through execSQL() method using the SQL statement to insert a record or use the insert() method passing some parameters like table name and ContentValues.

When the user does not enter anything but directly click the search button, popup a dialog that prompts user to enter a keyword as Figure 9-2.

Note: When exiting the program, you should shut down the database.

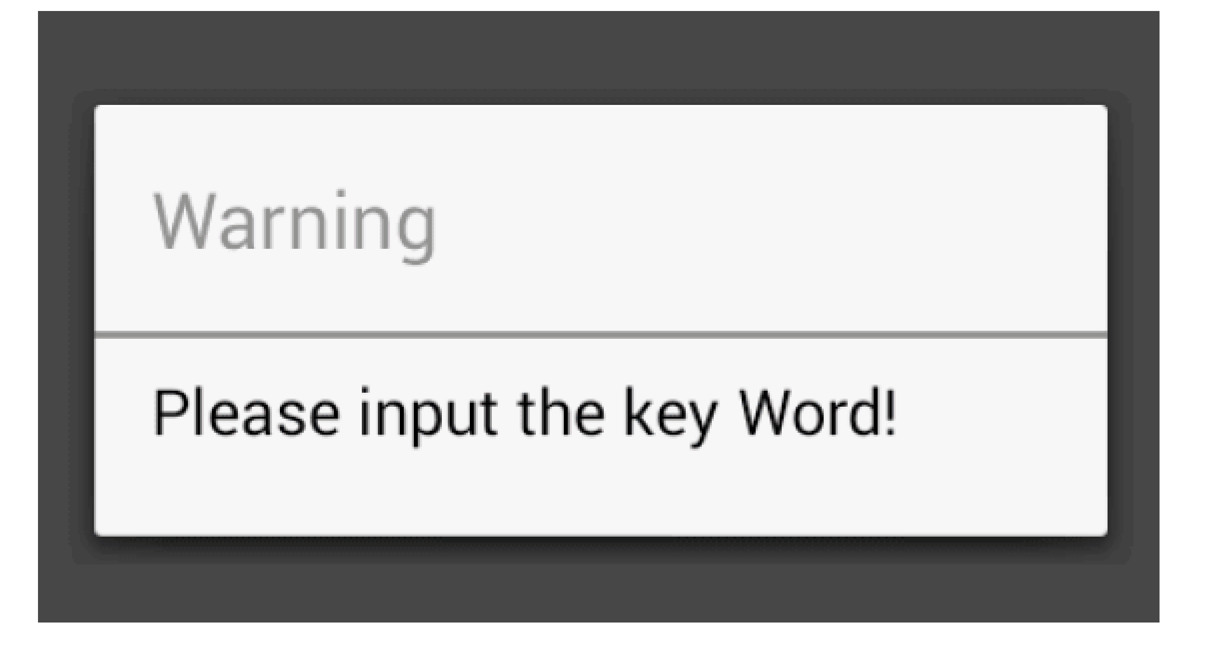


Figure 9-2 the figure of showing dialog

9.4 Extension of Knowledge

9.4.1 ArrayAdapter

ArrayAdapter is a subclass of BaseAdapter abstract class, which is usually used to store an array or a collection element. By default this class will display the content element on a TextView, if the content element is a object instead of a string, then will override the to String() method of your objects to determine what text will be displayed for the item in the list. To use something other than TextViews for the array display, for instance, ImageView. Then we need to override getView(int, View, ViewGroup) to return the type of view you want. Creating ArrayAdapter objects typically need to pass three parameters:

- **The current context**. Present Activity generally.
- **The resource ID for a layout file containing a TextView to use when instantiating views**. In this case android.R.layout.simple_list_item_1 is calling a layout file from system. This document contains only a label <TextView>. We can also simply create a layout file if we want to use a custom TextView, this layout needs a label <TextView> with various properties such as size, color and so on.
- **The objects to represent in the ListView.**

In addition to the commonly used three parameters, we can also pass a forth parameter. When the layout file has many labels, we need to specify the layout file, the ID of the TextView. (context,layout file, the ID of TextView widget, data resource)

9.4.2 Dialog

AlertDialog is a subclass of Dialog, it can create mostly dialog interacting with users. Also it is a recommender dialog, always used for notification, executed some small tasks related to the application. The steps to build an AlertDialog:

- Using builder object create AlertDialog object.
- Using SetMessage and SetTitle set the Body message and Title respectively.
- Using Create() method to create a AlertDialog.
- Using Show() method to display AlertDialog.

In the above steps, the inner class Builder class is mainly processed, and then we will see what kinds of methods are in it. The table below show the main method inside Builder class in Table 9-1.

Table 9-1 some methods for Builder class

Name	Function
public Builder setTitle	Set the title displayed in the Dialog.
public Builder setMessage	Set the message to display.
public Builder setIcon	Set the Drawable to be used in the title.
public Builder setPositiveButton	Set a listener to be invoked when the positive button of the dialog is pressed.
public Builder setNegativeButton	Set a listener to be invoked when the negative button of the dialog is pressed.
public Builder setNeutralButton	Set a listener to be invoked when the neutral button of the dialog is pressed.
public Builder setOnCancelListener	Sets the callback that will be called if the dialog is canceled.
public Builder setCancelable	Sets whether the dialog is cancelable or not.
public Builder setItems	Set a list of items to be displayed in the dialog as the content, you will be notified of the selected item via the supplied listener.
public Builder setMultiChoiceItems	Set a list of items to be displayed in the dialog as the content, you will be notified of the selected item via the supplied listener.
public Builder setSingleChoiceItems	Set a list of items to be displayed in the dialog as the content, you will be notified of the selected item via the supplied listener.
public AlertDialog create()	Creates an AlertDialog with the arguments supplied to this builder.
public AlertDialog show()	Creates an AlertDialog with the arguments supplied to this builder and shows the dialog.

Note : most methods in this table return a Builder, which means it returns to the Builder object itself after calling Builder object's method. We can understand the Initial Builder objects as an empty shell; each method of Builder is to add something to the specified location in this shell. Due to the location of each stuff is fixed, so the later value will cover the front value if there are multiple calls to the same method. The process of calling Builder object method is the process of creating a dialog, once built; it can be created and displayed.

9.5 Thinking and Exercises

(1) The list item style using in this case — android.R.layout.simple_list_item_1 is provided by system. Please customize a style that the popup text color is red, size 24sp and the distance between items is 20dp.

(2) Which attribute of AutoCompleteTextView can we use to set the number of characters

that the user must type before the completion suggestions are displayed in a drop down menu? (   )

A. android :completionThrehold   B. android :completionHint

C. android:dropDownVerticalOffset   D. android:dropDownHorizontalOffset

(3) Which option has no inheritance relationship between the two class? (   )

A. TextView, AutoCompleteTextView   B. TextView, Button

C. ImageView, ImageSwitcher   D. ImageView, ImageButton

(4) Which option is not inherited from BaseAdapter? (   )

A. ArrayAdapter   B. SimpleAdapter   C. CursorAdapter   D. PagerAdapter

(5) Which statement about AlertDialog is false? (   )

A. the method of AlertDialog show() can create and display a dialog

B. the method of AlertDialog.Builder create() and show() both return an AlertDialog object

C. AlertDialog cannot be constructed by the key word now; it must through the inside class Builder

D. the method of AlertDialog.Builder show() can create and display a dialog

(6) There are many methods in class Builder. Which return value type is different from others among the following methods? (   )

A. create()   B. setMessage()   C. setView()   D. setAdapter()

(7) The maximum number of button in an AlertDialog is (   ).

A. 0   B. 1   C. 2   D. 3

Chapter 10　Simulate Gallery

10.1　Case Overview

This case mainly introduced the gallery effect realized by the horizontal scroll bar and the horizontal LinearLayout widget. In the early versions of Android, the Android system provides the Gallery widget to realize this effect; it is a pity that the Gallery widget no longer exists in Android 4.1. Android officially recommends using the horizontal scroll bar or the ViewPager widget to realize this effect. This case using the horizontal scroll bar to realize the gallery effect when you click a picture at the bottom of screen, the picture will show the corresponding picture at the top of screen. The selected picture at the bottom list will be showed completely, while unselected picture will be showed translucently. Our application running looks like Figure 10-1.

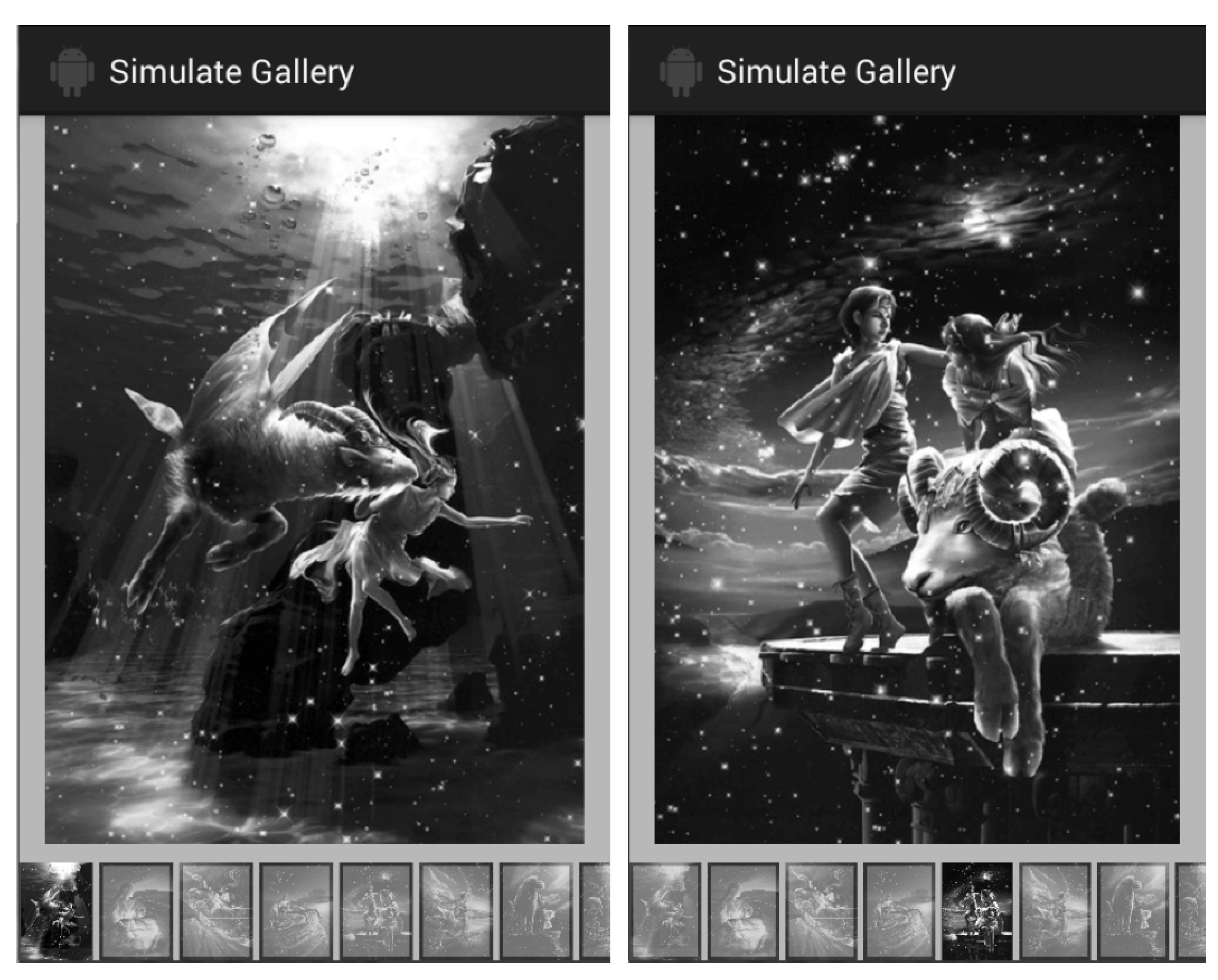


Figure 10-1　the figure of the running results at different times

10.2 Key Code

Layout file: 10\ScrollViewGallery\res\layout\activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc"
    android:orientation="vertical" >
    <ImageSwitcher
        android:id="@+id/mSwitcher"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:paddingBottom="10dp"
        android:inAnimation="@android:anim/fade_in"
        android:outAnimation="@android:anim/fade_out"/>
    <HorizontalScrollView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <LinearLayout
            android:id="@+id/mLinear"
            android:orientation="horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </LinearLayout>
    </HorizontalScrollView>
</LinearLayout>
```

Custom frame graphics files: 10\ScrollViewGallery\res\drawable-hdpi\bg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" >
    <solid android:color="#00000000"/>
    <stroke android:color="#ff0000"
            android:width="2dp"/>
</shape>
```

Program code: 10\ScrollViewGallery\src\iet\jxufe\cn\android\scrollviewgallery\MainActivity.java

```
public class MainActivity extends Activity {
    private LinearLayout mLinear;//to show thumbnail at the bottom
    private ImageSwitcher mSwitcher;
    List<Integer> imgIds;//collection to store image ids
```

```
    private ImageView[] imgViews;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mLinear = (LinearLayout) findViewById(R.id.mLinear);
        mSwitcher = (ImageSwitcher) findViewById(R.id.mSwitcher);
        //the initialization of the ImageSwitcher
        mSwitcher.setFactory(new ViewFactory() {
            public View makeView() {
                ImageView img = new ImageView(MainActivity.this);
                return img;
            }
        });
        imgIds=getImageIds();//get all images
        mSwitcher.setImageResource(imgIds.get(0));//display the first
                                                  //image by default
        init();//to perform initialization
    }
    public void init(){
        imgViews = new ImageView[imgIds.size()];//create a array to store
                                                //imageViews
        //create a layout parameters to specified size and margin
        LinearLayout.LayoutParams layoutParams=new
                                   LinearLayout.LayoutParams(60,80);
        layoutParams.setMargins(0, 0, 5, 0);//set the margin between the
                                            //images
        //create an ImageView for each image and carry on the simple Settings
        for (int i = 0; i < imgViews.length; i++) {
            imgViews[i] = new ImageView(this);//create an ImageView
            imgViews[i].setId(imgIds.get(i));//add id attribute
            imgViews[i].setBackgroundResource(R.drawable.bg);
            //set border for ImageView
            imgViews[i].setImageResource(imgIds.get(i));//set images
                                                        //for ImageView
            imgViews[I].setLayoutParams(layoutParams);
            imgViews[I].setOnClickListener(new MyListener());
            if(I!=0){//the first fully display by default, other
                     //translucent display
                imgViews[i].setImageAlpha(100);
            }else{
                imgViews[i].setImageAlpha(255);
            }
            mLinear.addView(imgViews[i]);//add ImageView to LinearLayout
        }
```

```
    }
    private class MyListener implements OnClickListener {
    //click event listeners for the ImageView at the bottom
        public void onClick(View v) {
            mSwitcher.setImageResource (v.getId());
            setAlpha(imgViews);
            ((ImageView) v).setImageAlpha(255);
        }
    }
    public void setAlpha(ImageView[] imageViews) {//set the Alpha of all
                                                   //the ImageView to 100
        for (int i = 0; i < imageViews.length; i++) {
            imageViews[i].setImageAlpha(100);
        }
    }
    //get all the eligible image ID by the reflection mechanism
    public List<Integer> getImageIds(){
        List<Integer> imageIds=new ArrayList<Integer>();
        try {
            //obtain all the fields of R.drawable class
            Field[] drawableFields = R.drawable.class.getFields();
            //to iterate these fields and then determine whether conform
  //to the specified conditions, if qualified, add its value to the collection
            for (Field field : drawableFields) {
                if (field.getName().startsWith("x_")) {
                    imageIds.add(field.getInt(R.drawable.class));
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return imageIds;
    }
}
```

10.3 Code Analysis

10.3.1 Interface analysis

It is clear the interface mainly contains two parts: The top part is a ImageSwitcher used to show the larger picture, the bottom part is a horizontal LinearLayout used to show all the thumbnail pictures. Because there are many pictures, the screen cannot accommodate all the

pictures, by default the widget beyond the screen cannot be displayed in the LinearLayout. In order to display all the pictures, we added a horizontal scroll bar outside the LinearLayout. You can see the picture beyond the screen by touching the scroll bar horizontally.

The numbers of the widgets are determined by the numbers of pictures, and the status of the ImageSwitcher is dynamically changed according to users' operation, it is hard to add and specify in layout file. The LinearLayout widget just specifies the direction of the LinearLayout and is temporarily empty. The codes is listed in activity_main.xml.

10.3.2 ImageSwitcher Introduction

ImageSwitcher is mainly used to switch and display pictures. The difference between the ImageSwitcher widget and the ImageView widget can be represented when the picture changes you can add an animated picture on the state entry and exit.

The ImageSwitcher creates two views for switching pictures using the setFactory() method. This method passing a parameter type is the ViewFactory, the parameter is a factory interface, especially used for creating widget. The interface just has one makeView() method, it will return to the created widget. If you want to implement the ViewFactory interface, you need to realize the makeView() method. As a picture switcher, the makeView() method should return to the widget-displayed pictures, the widget is the ImageView widget. In the setFactory(), it actually creates two ImageView widgets to switch the pictures by calling twice the makeView() method of ViewFactory interface .

Therefore, after creating the ImageSwitcher objects, you need to initialize it by calling its setFactory() method, otherwise you can not realize switching function. There are two methods adding an animated picture on the state entry and exit.

(1) In the layout file, add **android: inAnimation** and **android: outAnimation** attribute to set a animated picture on the state entry and exit. The animated picture can be the picture provided by the system and also can be the picture user-defined. Such as the effect of fadein and fadeout:

```
android:inAnimation="@android:anim/fade_in"
android:outAnimation="@android:anim/fade_out"
```

(2) In the Java code, calling the setInAnimation() and setOutAnimation() method of ImageSwitcher object to pass the corresponding animation. For example:

```
mSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,android.R.anim
.fade_in));
mSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,android.R.anim
.fade_out));
```

When you need to switch to the next picture, just pass a resource ID by calling the mSwitcher.setImageResource() method, and the ID resource is the picture will be showed.

10.4 Expansion of Knowledge

In the previous case of switching pictures, we store the pictures' entire ID in an array. When initialized to the array, you need to add each picture's ID, if there is a lot of pictures, the codes will be a lot. Especially add new pictures or modify the picture's name, we need to modify the codes, low extensibility and flexibility. In fact we can dynamically get the picture ID in Android through the Java reflection mechanism. Because the image files in the Android will generate the resource ID in the R.drawable class, and each image file corresponding to an element of the R.drawable class. We get all the elements of R.drawable class, which means we get all the pictures' ID. We can get the qualified picture by judging the variable member name. For Java reflection mechanism, it is easy to get the name of variable member.

So-called reflection means in the running state, for any class, you are able to know all the properties and the methods of the class; for any object, you can call any methods and properties of the object. The key of the Java reflection mechanism is to get what you want to explore the Class. After having a class object, you can further access to the variable members, methods, constructors of the Class. The related API is stored in the Java.lang.reflect package. There are three methods to get the Class object in Java code:

(1) Use the static method forName() of the Class class. This method need to pass a parameter of string type, the string contains complete package name + class name;

(2) Call the class attribute of a class to gain the corresponding class object, in this case, use the R.d rawable. The class will return to the class object corresponding with the R.drawable class;

(3) Call the getClass() method of an object, which is a method of java.lang. Object class, all classes can call this method, which will return the corresponding Class object.

After getting the Class object corresponding with the R.drawable class. You can call the getFields() method to obtain all public variable members in R.drawable class, call the getName() method of the Field object's to get the name of variable member the picture's file name, and then determine whether is the picture we want, if it is, then get the value of the variable member means picture ID.

10.5 Thinking and Exercises

(1) Please modify the existing program, add two Buttons around ImageSwitcher widgets, click the Button, you can view a picture in preview or next, at the same time the thumbnail

pictures also changed.

(2) Which class is not stored in java. lang. reflect package?( )

A. Class B. Field C. Method D. Constructor

(3) Defined a LinearLayout, how to add the widget in the layout?( ).

A. addAction B. addView C. addChild D. addLay

Chapter 11　Android Books List

11.1　Case Overview

This case mainly introduced the usages of the ListView widget. When the application contains a number of data, and each data structure is the same, just the content is different; usually we use the ListView to show all of them. The contents of the list item may be a very simple TextView displayed strings, or may be a complicated container with multiple widgets. In this case, the list shows the basic information of books, and click an item of the list, you can see the detailed informations of the book, then the image spread out. This case involves the construction of a complex list item, event handling of list item, the page jumps, data transfer, etc. Our application running looks like Figure 11-1.

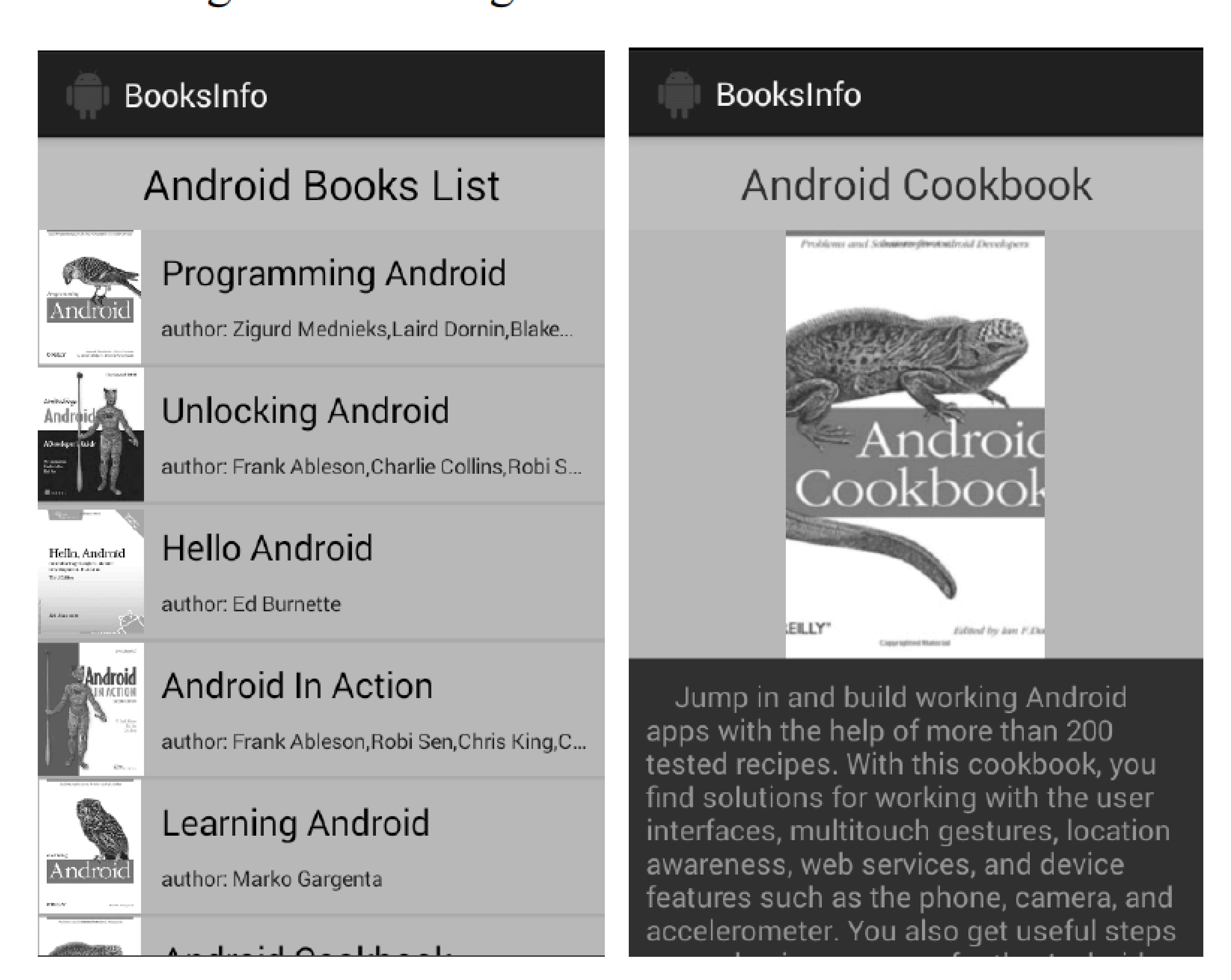


Figure 11-1　the figure of the running results in different pages

11.2　Key Code

The main interface layout file: 11\BooksInfo\res\layout\activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc"
    android:orientation="vertical">
    <TextView
        android:layout_width=" wrap_content "
        android:layout_height="wrap_content"
        android:text="@string/title"
        android:textSize="24sp"
        android:background="#ccbbaa"
        android:textColor="#000000"
        android:padding="10dp"
        android:gravity="center" />
    <ListView
        android:id="@+id/mBooks"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:divider="#aaaaaa"
        android:dividerHeight="2dp"
        android:gravity="center" />
</LinearLayout>
```

Each item of list layout file: 11\BooksInfo\res\layout\item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/image"
        android:layout_width="60dp"
        android:layout_height="75dp"
        android:contentDescription="@string/imgInfo" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
            android:textColor="#000000"
            android:textSize="20sp" />
        <TextView
            android:id="@+id/author"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:gravity="left"
            android:singleLine="true"
            android:ellipsize="end"
            android:textColor="#0000ee"
            android:textSize="12sp" />
    </LinearLayout>
</LinearLayout>
```

detail information layout file: 11\SceneryInfo\res\layout\scenery_show.xml

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#aabbcc"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:padding="10dp"
            android:background="#ccbbaa"
            android:textSize="24sp"/>
        <ImageView
            android:id="@+id/image"
            android:layout_width="200dp"
            android:layout_height="240dp"
            android:scaleType="fitXY"
            android:contentDescription="@string/imgInfo"/>
        <TextView
            android:id="@+id/content"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#333333"
            android:textColor="#00ff00"
```

```
            android:textSize="16sp"
            android:padding="10dp"/>
    </LinearLayout>
</ScrollView>
```

Program code: 11\SceneryInfo\src\iet\jxufe\cn\android\sceneryinfo\MainActivity.java

```
public class MainActivity extends Activity {
    private ListView mBooks;
    //the collection to save every book information
    private List<Map<String, Object>> list = new ArrayList<Map<String,
Object>>();
    private int[] imgIds = new int[] { R.drawable.programming_android,
            R.drawable.unlocking_android, R.drawable.hello_android,
            R.drawable.android_in_action, R.drawable.learning_android,
            R.drawable.android_cookbook };
    private String[] titles = new String[] { "Programming Android",
            "Unlocking Android", "Hello Android","Android In Action",
            "Learning Android","Android Cookbook"};
    private String[] authors = new String[] { "Zigurd Mednieks,Laird
            Dornin,Blake Meike", "Frank Ableson,Charlie Collins,
            Robi Sen","Ed Burnette","Frank Ableson,Robi Sen,Chris King,
            C. Enrique Ortiz","Marko Gargenta","Ian F. Darwin" };
    private int[] contentIds = new int[] { R.raw.programming_android,
            R.raw.unlocking_android,
            R.raw.hello_android, R.raw.android_in_action,
            R.raw.learning_android,
            R.raw.android_cookbook};
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mBooks = (ListView) findViewById(R.id.mBooks);
        Init();//initializtion, put the book information in a Map Object
        //Create Adapter Object, contact data source with the listview
        SimpleAdapter adapter = new SimpleAdapter(this, list,
              R.layout.item,
                new String[] { "img", "title", "author" }, new int[] {
                        R.id.image, R.id.title, R.id.author });
        mBooks.setAdapter(adapter);
        //add click Listener for listView
        mBooks.setOnItemClickListener(new OnItemClickListener() {
            //handling the click operation, go to another page and
            //transfer data
            public void onItemClick(AdapterView<?> parent, View view,
                    int position, long id) {
```

```
                    Intent intent = new Intent();
                    intent.setClass(MainActivity.this,
                                    BookInfoShowActivity.class);
                    intent.putExtra("title",(String)
                                    list.get(position).get("title"));
                    intent.putExtra("image", (Integer)
                                    list.get(position).get("img"));
                    intent.putExtra("content",(Integer)
                                    list.get(position).get("content"));
                    startActivity(intent);//go to another page
                }
            });
        }
    private void init() {
        //initializtion, put the book information in a Map Object, and
        //put all books information in a list
        for (int i = 0; i < imgIds.length; i++) {
            Map<String, Object> item = new HashMap<String, Object>();
            item.put("img", imgIds[i]);
            item.put("title", titles[i]);
            item.put("author","author: "+authors[i]);
            item.put("content", contentIds[i]);
            list.add(item);
        }
    }
}
```

Program code: 11\BooksInfo\src\iet\jxufe\cn\android\booksinfo\BookInfoShowActivity.java

```
public class BookInfoShowActivity extends Activity {
    private TextView title,content;//the TextView to display title and
                                    //content
    private ImageView imageView;//the ImageView to display the Image
    private ClipDrawable clipDrawable;//used to  spread image slowly
    private Handler mHandler;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.scenery_show);
        imageView=(ImageView)findViewById(R.id.image);
        content=(TextView)findViewById(R.id.content);
        title=(TextView)findViewById(R.id.title);
        //get the passing data, title, the id of image and the id of detail
        //information files
        String titleStr=getIntent().getStringExtra("title");
        int image=getIntent().getIntExtra("image",
```

```
                                       R.drawable.programming_android);
        int info=getIntent().getIntExtra("content",
                                            R.raw.programming_android);
        title.setText(titleStr);
        //Create a ClipDrawable object, specified the image and direction
        //of interception
        clipDrawable = new ClipDrawable(getResources().getDrawable(
                image), Gravity.CENTER, ClipDrawable.HORIZONTAL);
        imageView.setImageDrawable(clipDrawable);
        //start the thread, begin to intercept image
        startThread();
        //according to the id to get the input stream
        InputStream inputStream=getResources().openRawResource(info);
        content.setText(getStringFromInputStream(inputStream));
        mHandler=new Handler(){//create a handler object.
            public void handleMessage(Message msg) {//handler message
                clipDrawable.setLevel(clipDrawable.getLevel()+400);
                 //to increase the displayed part of image
            }
        };
    }
    //to read the string from the input stream
    public String getStringFromInputStream(InputStream inputStream){
        byte[] buffer=new byte[1024];
        int hasRead=0;//record the number of bytes has read
        StringBuilder result=new StringBuilder("");
        try{
            while((hasRead=inputStream.read(buffer))!=-1){
                //construct a string with bytes and added to the existing
                //string
                result.append(new String(buffer, 0,hasRead,"GBK"));
            }
        }catch (Exception ex){
            ex.printStackTrace();
        }
        return result.toString();
    }
    public void startThread(){
        clipDrawable.setLevel(0);//set the picture invisible at
                                    //initialization
        new Thread(){//create a thread
            public void run() {
                while(clipDrawable.getLevel()<10000){
                        //jude the picture whether had fully displayed
```

```
                    try {
                        Thread.sleep(300);//sleep 0.5 second
                        mHandler.sendEmptyMessage(0x11);
                                                 //send an empty message
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }
        }.start();//start thread
    }
}
```

11.3 Code Analysis

11.3.1 Interface analysis

It is clear there are two parts interface in this case, one is used to display the information of books, the other one is used to display the detail information of the books when you click one item at the main interface, then you can get to the corresponding page which displayed the detail information.

In the main interface, it mainly contains two widgets, one is the TextView widge used to display the title, the other one is the ListView used to display the detail information of book, and the whole adopt vertical LinearLayout. In the ListView, set the color of the divider line between the items and the height of the line, android:divider, android:dividerHeight, the code listed in the activity_main.xml.

Each item in the list contains three parts of information: picture of book, book name, introduction, these three parts of information integrated in a horizontal LinearLayout, nested with a vertical LinearLayout, as shown below in Figure 11-2. Specific code in the item.xml:

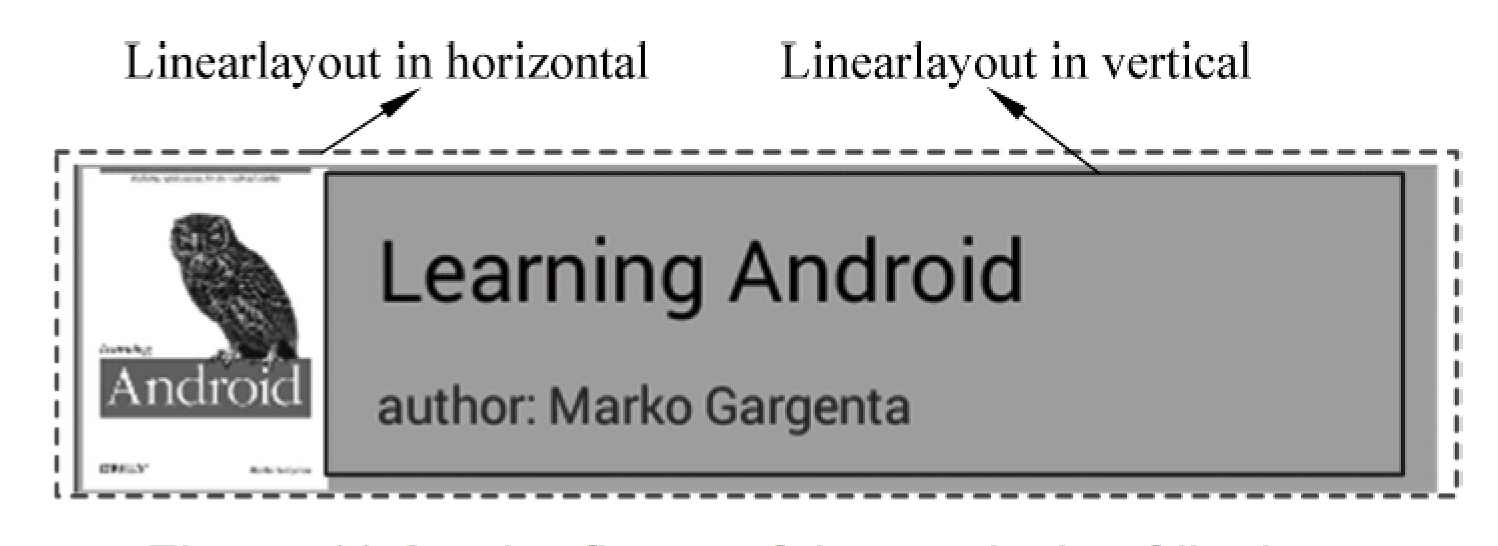


Figure 11-2 the figure of the analysis of list item

The interface display detail information contains three widgets: a TextView widget displaying title, an ImageView widget displaying picture, a TextView widget displaying the information of book. Overall adopt vertical LinearLayout, the specific code in the scenery_show.xml.

Note: When you create a new activity, you must register it in the manifest.xml in advance.

11.3.2 ListView

ListView is the most common widget in android, displays all items in a vertical list form. In Android, using ListView seems like using other widgets; usually adding the label of ListView in the layout file, then getting the ListView widget in code according to the ID. So that you can set its data, etc. In addition, the Android also provided the ListActivity, includes a ListView widget, so you can get a ListView widget by letting this activity inherit the ListActivity, do no need the layout file.

After get the ListView widget, the key point is to specify a data source for it. However the ListView cannot be associated with the data source directly, it needs help of Adapter. Then through the setAdapter()method associated Adapter with the ListView. The Adapter mainly used in associated the data source with the widget. The Hierarchical structure of adapter as followed in Figure 11-3:

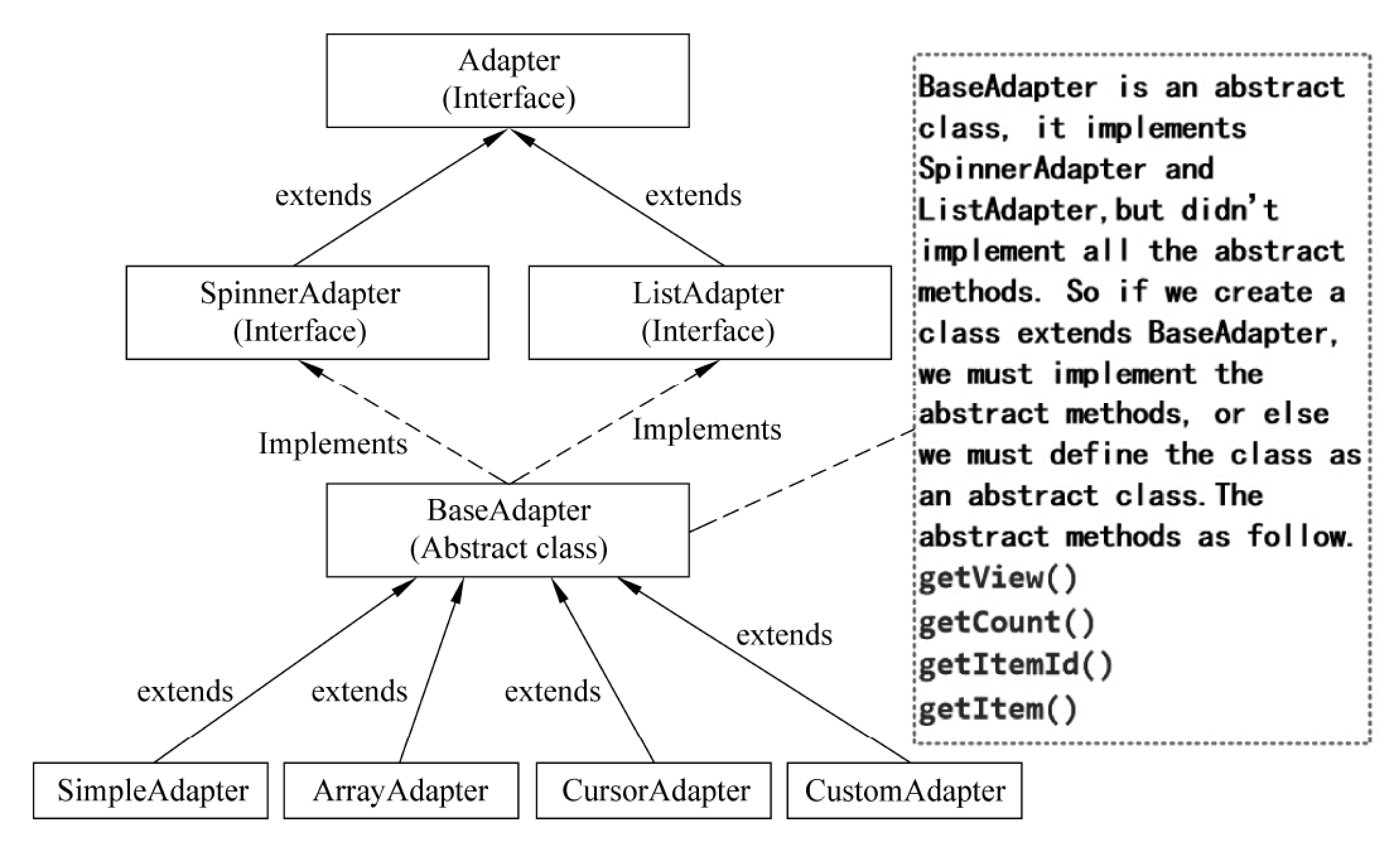


Figure 11-3 hierarchical structure diagram of Adapter

The BaseAdapter class is the most important class in the Adapter, the class itself is an abstract class, it cannot be instantiated, but it is the base class for the Adapter class, developers can just inherit from this class, then override the abstract method, do all of those you can create a custom Adapter. Creatig a custom Adapter must implement the following methods:

- getView(): Get a View displays the data at the specified position in the data set. You can either create a View manually or inflate it from an XML layout file. When the View is inflated, the parent View (GridView, ListView...) will apply default layout parameters unless you use inflate(int, android.view.ViewGroup, boolean) to specify a

root view and to prevent attachment to the root;

- getCount(): How many items are in the data set represented by this Adapter.;
- getItemId(): Get the row id associated with the specified position in the list;
- getItem(): Get the data item associated with the specified position in the data set.

The most important method is the getView() method and the getCount() method, through the getCount() method you can know how many data is listed in the adapter.You can get each data and add them to the ListView to form a list contains data through the getCount() method recursive call the getView() method.

Through the custom Adapter, we can put the data displayed in any form just we want, also we can add event processing for some items, but the disadvantage is that we need to override multiple methods, then there will be a lot of codes. In considered of this, the Android system provided some common subclass ArrayAdapter, SimpleAdapter, CursorAdapter, etc. These classes have their own characteristics when used in suitable places. For example, the ArrayAdapter is suitable for the list only contains text, SimpleAdapter is suitable for the list with complex structure, and CursorAdapter is suitable for changing the database query results to the list.

11.3.3 SimpleAdapter

SimpleAdapter is a simple and practical adapter, it can associate the static data with the widget in xml file. Usually we conserve the static data by putting them into a collection of Map objects, a map object corresponding to the item, which in the list contains all the data, through the keyword of a map object we can identify each kinds of data. For example, in this case, a book contains four parts data: the picture of book, the name of book, the introduction, the detail information of book, so the map object contains four keywords, see the codes 38th -47th line, a collection of all the map objects formed the data source.

After we get the data source, how does the data displayed? The data is generally displayed on the corresponding widgets, so we need to define a layout file for each single item. The layout file can be really easy with one widget, it can also be very complex, consisting of multiple layout nested in it. In this case, each item need to display three parts of information, the layout file has three widgets and specify ID for each widget. Specific code in the item.xml.

Now, we had the data source and the corresponding widgets, the key point is how to associate the data item in the data source with the widget in the layout file, and make the data displayed in right ways. We use the keyword to specify the corresponding value of each part in the data in the map object. In the layout file, we use the ID to specify the widget. We build individual related relationship between the keyword of map object and the ID of widget in the layout file, which will help us to keep data consistency.

With above analysis we can easily understand the construction method of the SimpleAdapter class, SimpleAdapter(Context context, List<? extends Map<String, ?>> data,

int resource, String[] from, int[] to).That means when you create SimpleAdapter object,need to pass five parameters.

- context The context where the View associated with this SimpleAdapter is running;
- data A List of Maps. Each entry in the List corresponds to one row in the list. The Maps contain the data for each row, and should include all the entries specified in “from”;
- resource Resource identifier of a view layout that defines the views for this list item. The layout file should include at least those named views defined in “to;”;
- from A list of column names that will be added to the Map associated with each item;
- to The views that should display column in the “from” parameter. These should be Text Views. The first N views in this list are given the values of the first N columns in the from parameter.

Note: There is one to one correspondence between the fourth parameter and fifth parameters, according to the data collected by the fourth parameter, it will be displayed in the fifth parameter widget, and the element in the fifth parameter must be in the layout file which specifies in the third parameter.

11.3.4 ClipDrawable

ClipDrawable means cutting out a picture fragment from a bitmap. ClipDrawable can realize the effect pictures slowly unfolding effect, the principle is: repetition of intercept the same picture, but each time the interception of fragment is different size, initially the interception of the fragment is very small, and gradually growing, until the interception of the entire image, this appears start slowly.

ClipDrawable object can be created in code, can also be defined in the XML file. When you define the ClipDrawable object,use the <clip.../> element be the root element, it mainly contains the following attributes:

- android:drawable: Reference to a drawable resource to draw with the specified scale.
- android:clipOrientation: The orientation for the clip.
- android:gravity: Specifies where to clip within the drawable.

When directly creating ClipDrawable object in code, ClipDrawable (Drawable drawable, int gravity, int orientation) need to pass three parameters, the first one is the picture need to intercept, the second one is the direction of interception, the third one is the alignment of interception.

When you use the ClipDrawable object intercept pictures, through the setLevel (int level) method to set the size of the interception, when the level value is 0, the interception area is empty; when the level value is 10000, the interception area is the whole picture. This case is to start a thread to determine whether the interception is the whole picture, if not, every 0.3 seconds the system will send a request. And in each interception, the value of level will be

increased by 400.

11.4 Extension of Knowledge

11.4.1 The raw directory

In this case, the detail information of books is abundant; it should not be placed in the code, putting the introduction of the books in a separate TXT file. Then we create a raw folder in the res directory, and put the TXT files into the raw folder.

The raw folder is the folder used to storing the original android resource files, it keeps the files wholly intact stored on the device, and those files will not be compiled into binary form. Through the R.raw, You can open and read its content by using getResources(), openRaw Resources(R.raw.XXX) through R.raw.XXX reference files. But you cannot create a subfolder under this folder.

In the Android applications, the assets folder and the res folder is also used for storing resource, what is the difference between them? Table 11-1 shows some important differences.

Assets folder: Used to store the static files, which need to be packed in the setup program, the asset folder will keep the resources stored here wholly intact, so they will not compiled into a binary. Unlike res folder, asseting folder supports arbitrary depth in sub directory means you can create many sub folders just like you are in the folder. These files will not generate any resource flags, you must use the /assets to (but it does not contain it) begin the relative path name, you need to use the AssetManager class to access the assets folder, if you want to read, you should use the file stream.

res: Used to store the application's resources(such as icons,interface layout), it will generate markers in the R.java file, the resources will be packaged into package, those unused resources will not be packaged into package. This folder contains some fixed sub folders, you can not create sub folders at will.

Table 11-1 the differences of three file folders

Compare item	assets folder	res folder	res/raw folder
Whether generating resource markers in R.java	NO	YES	YES
Whether can create sub folder	CAN	CAN NOT	CAN NOT
Whether will be compiled into a binary file	NO	YES	NO
Whether completely pack package into the installation file	YES	NEED JUDGMENG	NEED JUDGMENT
Access mode	Asset Manager class, through file stream	Through the R.raw.XXX reference file	Through the R.raw.XXX reference file

11.4.2 Activity overview

An Activity is an application component providing a screen and the users can do a lot of things, such as dial the phone, take a photo, send an email, or view a map etc.

We usually use a website as an analogy for activities. Just like a website consists of multiple pages, so does an Android application consist of multiple activities. Just like a website has a “home page,” an Android app has a “main” activity, usually the one that is shown first when you launch the application. And just like a website has to provide some sort of navigation among various pages, an Android app should do the same performance. For example, when the user starts an application for the first time, the Activity Manager will create its activity and put it onto the screen. Later, when the user switches screens, the Activity Manager will move the previous activity to a holding place. In this way, if the user wants to go back to an older activity, it can be started more quickly. Older activities that the user hasn’t used in a while will be destroyed in order to free more space for the currently active one. This mechanism is designed to help improve the speed of the user interface and thus improve the overall user experience.

If you want to create your own activity, you must inherit from the base activity class or the existing activity subclass, such as ListActivity etc. You can realize the callback method in your activity defined by the system activity class, these callback method will be automatically called by the system when your activity status is going to be changed, one of the most important callback method is the onCreate() method.

When the activity is created, the system will automatically callback its onCreate() method. In the implementation of the method, you should initialize some key widgets. The most important is to call the setContentView() method to set the corresponding layout file in your activity. In order to control the widget in the interface, you can call the findViewById(int id) method in Activity to get the widget, then you can modify the widgets’attribute and the called method.

After defining your activity, the system also cannot access the activity until you have been registered and configured in the Android Manifest.XML file. In the previous program, we also have our own activity, however we did not configure it, we can access it,right? Because, all the previous program examples just have one activity, our development tools has automatically configured it at the time of its creation, and use it as the main activity, the default configuration as follows.

```
<activity
    android:name="iet.jxufe.cn.android.sceneryinfo.MainActivity"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Holo.Light.NoActionBar" >
    <intent-filter>
```

```
        <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

The content between the <intent-filter> element and </intent-filter> element shows the activity is the main activity.

When we defined our activities, we just add sub tab <activity.../>in the label <application.../> element, then set its associated properties, mainly as follows.

name: The name of the class that implements the activity, a subclass of Activity. The attribute value should be a fully qualified class name (Such as, “com.example.project. Extracurricular Activity”);

icon: An icon representing the activity. The icon is displayed to users when a representation of the activity is required on-screen. For example, icons for activities that initiate tasks are displayed in the launcher window. The icon is often accompanied by a label;

label: A user-readable label for the activity. The label is displayed on-screen when the activity must be represented to the user. It’s often displayed along with the activity icon.

In addition, configurate activity can also specify one or more <intent-filter...>element, the element is used to specify the conditions which the activity response to.

In the configuration, only the name attribute is required, while other properties or tag elements is optional.

After the activity has been created and configured, you can use the startActivity (Intent intent) method to start activity, this method need to pass a parameter which type is intent. The parameter is the description of your waiting started activity, it can be an exact activity class or the characteristics of activity you want to start, then the system will search the activity with such features. If there are multiple activities to meet the requirement, the system will list a list displaying all the activities, let the user choose which one is to startup.

In this case, we can use the intent.set Class(Main Activity.this, Scenery Show Activity.class) method to specify the activity which one is needed to startup. And also can pass some data to the started activity, the data stored in the intent object. You can get the values by relevant keys in the target activity for the object stored data in the form key value pairs.

11.5 Thinking and Practice

(1) In this case, the effect is horizontal, starting from the middle to both sides, spread out slowly, please try to modify the code, making it spread out slowly from top to the bottom.

(2) Write a class inherits from BaseAdapter, which method is no need to write again?(   )

A. getCount()   B. getView()   C. getItem()   D. getDropDownView()

(3) In android,there are a lot of related adapter classes, the following options,which one is

not inherited from BaseAdapter?( )

A. ArrayAdapter B. SimpleAdapter C. CursorAdapter D. PagerAdapter

(4) Which one of the following description of parameter in SimpleAdapter is not correct?( )

A. The first parameter is the context object, usually just pass the current object will be fine

B. The second parameter is the data source of list, which can be an array, or a set

C. The third parameter is the layout file of each item in the list, the layout contains some widgets

D. There is one to one correspondence between the fourth parameter and fifth parameter, according to the data which get by the fourth parameter, it will be displayed in the fifth parameter widget, and the element in the fifth parameter must in the layout file which specified in the third parameter

(5) When configured the activity ,which one is the essential one? ( )

A. android:name B. <action…/> C. <intent-filter…/> D. <category…/>

Chapter 12　BBC News—ListView Delay Load

12.1　Case Overview

This case is focused on the usage of ListView and displays news through a list, which is very similar to the previous application Android Books List. Each news contains three pieces of information including picture, title and brief introduction. But there are two differences between them:

(1) All news and information are stored in the database instead of specified in the code directly.

(2) The news list does not display all the news data immediately, but show part of the news record only. When the user slides to the last news, the bottom information is being loaded, then load the new record and add to the list. This case involves how to build a complex list of items, operate the database and scroll event handling, etc. Our application running looks like Figure 12-1.



Figure 12-1　the figure of the running results at different times

12.2 Key Code

Main interface layout file: 12\ListViewDelayLoad\res\layout\activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ccbbaa"
    android:orientation="vertical" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:padding="10dp"
        android:text="@string/title"
        android:textColor="#000000"
        android:gravity="center_vertical"
        android:drawableLeft="@drawable/logo"
        android:textSize="24sp" />
    <ListView
        android:id="@+id/news"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:divider="#aaaaaa"
        android:dividerHeight="2dp"
        android:background="#aabbcc"
        android:gravity="center" />
</LinearLayout>
```

Layout file of each item in the news list: 12\ListViewDelayLoad\res\layout\item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/image"
        android:layout_width="75dp"
        android:layout_height="60dp"
        android:scaleType="fitXY"
        android:layout_margin="10dp"
```

```
        android:contentDescription="@string/imgInfo" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:paddingRight="5dp"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="#000000"
            android:singleLine="true"
            android:ellipsize="end"
            android:textSize="18sp" />
        <TextView
            android:id="@+id/info"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="left"
            android:textColor="#0000ee"
            android:textSize="14sp"
            android:maxLines="2"
            android:layout_marginTop="5dp"
            android:ellipsize="end"/>
    </LinearLayout>
</LinearLayout>
```

Layout file of loading information at the bottom: 12\ListViewDelayLoad\res\layout\load.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="horizontal" >
    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:text="@string/load"
        android:textSize="20sp"
        android:textColor="#0000ff"
        android:gravity="center_vertical"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"/>
</LinearLayout>
</LinearLayout>
```

Program code: 12\ListViewDelayLoad\src\iet\jxufe\cn\android\listviewdelayload\MainActivity.java

```
public class MainActivity extends Activity {
    private ListView news;//ListView to display news
    private SimpleAdapter simpleAdapter;//adapter associated with data
                                        //source and ListView
    //collection used to store news data have loaded
    private List<Map<String, Object>> newsList = new ArrayList
       <Map<String, Object>>();
    private MyOpenHelper mHelper;
    private SQLiteDatabase mDB;
    private int totalCount;//the total number of news, including loaded
                                 //and unload
    private int loadedCount;//the number of news have loaded
    private int lastItem;//the index of the last visible item
    private View footView;//view to display loading information at the bottom
    private int loadItemNum = 4;//the number of items for each loading
    private Handler myHandler = new Handler();
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);//hide the title
                                                       //bar
        setContentView(R.layout.activity_main);
        mHelper = new MyOpenHelper(this, "news.db", null, 1);
        mDB = mHelper.getReadableDatabase();//get database
        news = (ListView) findViewById(R.id.news);
        //convert the layout file into a View widget and add it to the
        //ListView
        footView = getLayoutInflater().inflate(R.layout.load, null);
        news.addFooterView(footView);
        newsList = getData("select * from news_tb order by _id limit 0,6",
null);
        simpleAdapter = new SimpleAdapter(this, newsList, R.layout.item,
                new String[] { "image", "title", "info" }, new int[] {
                        R.id.image, R.id.name, R.id.info });
        news.setAdapter(simpleAdapter);
        totalCount = getCount();
        //create a scroll event listener
        MyOnScrollListener myListener = new MyOnScrollListener();
        //add a scroll event listener for the ListView
        news.setOnScrollListener(myListener);
```

```
    }
    private class MyOnScrollListener implements OnScrollListener {
        public void onScroll(AbsListView view, int firstVisibleItem,
                int visibleItemCount, int totalItemCount) {
            lastItem = firstVisibleItem + visibleItemCount - 1;
            loadedCount = simpleAdapter.getCount();//record the number
                                                   //of the loaded items
            if (loadedCount >= totalCount) {
                    //If all items have loaded, the loading information
                    //will be no longer displayed
                news.removeFooterView(footView);
            }
        }
        public void onScrollStateChanged(AbsListView view, int scrollState) {
            //footView is also an item, when footView is displayed, the
            //value of lastItem equals the value of loadedCount
            if (lastItem == loadedCount
                    && scrollState == OnScrollListener.SCROLL_STATE_IDLE) {
                if (loadedCount <= totalCount) {
                 //if the items are not totally loaded, load new items
                    myHandler.postDelayed(new Runnable() {
                                              //execute after 3 seconds
                        public void run() {
                            if (loadedCount + loadItemNum > totalCount) {
                                loadMore(totalCount - loadedCount);
                                //load the residual datas
                            } else {
                                loadMore(loadItemNum);
                                //load count pieces of data by default
                            }
                        }
                    }, 3000);
                }
            }
        }
    }
    public void loadMore(int num) {
        List<Map<String, Object>> list = getData(
                "select * from news_tb order by _id limit ?,?", new String[]{
                        loadedCount + "", num + "" });
        newsList.addAll(list);//add the loaded news to the news collection
        simpleAdapter.notifyDataSetChanged();
         //update and display the ListView
```

```
    }
    //get news data according to the query statement
    public List<Map<String, Object>> getData(String sql, String[] args){
        List<Map<String, Object>> list = new ArrayList<Map<String, Object>>();
        Cursor cursor = mDB.rawQuery(sql, args);//query the qualified
                                                 //record
        while (cursor.moveToNext()) {
            //iterate each record, get the appropriate data and save the
            //data to the collection
            Map<String, Object> item = new HashMap<String, Object>();
            item.put("image",cursor.getString
                      (cursor.getColumnIndex("image")));
            item.put("title", cursor.getString(cursor.getColumnIndex
("title")));
            item.put("info", cursor.getString(cursor.getColumnIndex
("info")));
            list.add(item);
        }
        return list;
    }
    //query the number of records in the database
    public int getCount() {
        Cursor cursor = mDB.rawQuery("select count(*) from news_tb", null);
        if (cursor.moveToNext()) {
            return cursor.getInt(0);
        }
        return 0;
    }
    protected void onDestroy() {
        if(mDB!=null){
            mDB.close();
        }
        super.onDestroy();
    }
}
```

Database helper class: 12\ListViewDelayLoad\src\iet\jxufe\cn\android\listviewdelayload\MyOpenHelper.java

```
public class MyOpenHelper extends SQLiteOpenHelper {
    public String createTableSQL="create table if not exists news_tb" +
            "(_id integer primary key autoincrement,image,title,info)";
                                              //Create table statement
    //Constructor method
```

```
    public MyOpenHelper(Context context, String name, CursorFactory factory, int version) {
        super(context, name, factory, version);
    }
    //the method invoked when created, execute initialization
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(createTableSQL);
        //Initialization insert statement is omitted here.
    }
    //the method invoked when the version of database is updated
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    System.out.println("version changed:"+oldVersion+"-->"+newVersion);
    }
}
```

12.3 Code Analysis

12.3.1 ListView lazy loading principle

In order to improve the efficiency and performance of ListView, the ListView has a lot of items; Android applications will not load all the information of the ListView at one time, taking a batch loading strategy instead. With the slide of users, it is necessary to load needed data from the background dynamically and add the data to the ListView. This can greatly improve the application’s performance and user’s experience.

Specific action: when initialize ListView, pre-loaded several records, when the user slides to the last news, the bottom information is being loaded, then load multiple record from the background and add to the list. After the process of loading, the next step is to update the ListView. During this process we need to define some variable:

- **totalCount**: The number of all the news in database;
- **loadedCount**: The number of news which has been loaded into the list already;
- **loadItemNum**: The number of new loaded news each time;
- **lastItem**: The last visible item’s serial number in the list.

In this case, the widget shows loading message and it was not displayed at the bottom directly, but displayed with the scrolling of ListView, located at the end of ListView and which belonged to the ListView. Actually it is also an item of ListView, but it is rather special. Its structure is different from other ordinary items, no matter when adding it to the ListView, it is always the last item of ListView, we can add it to ListView by calling method addFooterView().

The essence of delaying load is according to the scroll state of ListView and determine should we load the additional data or not. Therefore, the ListView scrolling state needs to be monitored, and add a scroll event listener for the ListView. Implement the listener need to override two methods: onScroll() and onScrollStateChanged().

onScroll(AbsListView view, int firstVisibleItem,int visibleItemCount, int totalItemCount) is used to monitor scroll events. The method takes four parameters, the view represents the scroll widget, in this case it is the ListView. The firstVisibleItem represents the first visible item's number, visibleItemCount represents the total visible item numbers on the widget; totalItemCount represents the total item on the widget that including the load items at the bottom. The quantitative relation between lastItem, firstVisibleItem and visibleItemCount is last Item = firstVisibleItem + visibleItemCount – 1. The reason why minus 1 is that serial number begins at 0. There is no need to show message is loading at the bottom when all the visibleItemCount had counted the firstVisibleItem. So it is necessary to judge whether the value of loaded Count is greater than or equal total Count in onScroll() method. The onScroll codes: from 36th-43th line.

onScrollStateChanged(AbsListView view, int scrollState) is used to monitor the scroll bar status change event. The method takes two parameters, the view represents the scroll widget and scrollState represents the status of scroll bar. Generally we check whether the scroll bar can be rolled which means the status of scroll bar become OnScrollListener.SCROLL_STATE_IDLE. There are two situations when the scroll bar cannot be rolled: scroll down to the last one and scroll up to the first item. What we need to handle is the first situation, so we also need to judge whether lastItem equals loadedCount. Then check whether there is remainder data to load. If loadedCount<totalCount means there is remainder data to load. By default, we load four records each time. If the remainder items are less than 4, load all of them. The onScrollStateChanged codes: from 64th-81th line.

12.3.2 Introduction of SQLite database

SQLite database is an embedded lightweight relational database in Android. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indic, triggers, and views, contains in a single disk file. It contains all the function of the local data operation, easy use, and quick response.

SQLite uses dynamic typing. Content can be stored as INTEGER, REAL, TEXT, BLOB, or as NULL. It does not enforce data type constraints. Any data can be inserted into any column. For example, you can put arbitrary length strings into integer columns; float point numbers in Boolean columns, or dates in character columns. The data type you assign to a column in the CREATE TABLE command does not restrict what type of data can be put into the column so we can omit the type declarations behind the data column. In this case, the

CREATE TABLE command is used to create table if not exists news_tb (_id integer primary key autoincrement,image,title,info). There is one exception: Columns of type INTEGER PRIMARY KEY may only hold a 64-bit signed integer. An error will result if you try to put anything other than an integer into an INTEGER PRIMARY KEY column.

SQLite database supports most SQL92 syntax and also allows developers to use SQL statements to manipulate data in the database. Common examples of standard SQL statements:

SELECT: select * from table where condition statement is the group-by-group statement having ... order by order statement.

e.g. select * from person order by id desc

Query all the records in table person descending order by id number.

select name from person group by name having count(*)>1

Query all the records which name field has appeared more than once in table person.

PAGING: select * from table limit skipped records, displayed records

e.g. select * from person limit 3,5

Get five records from the person table, skipping the previous three records.

INSERT: insert into table(field) values(values)

e.g. insert into person(name, age) values('Sam',26)

Insert a record to the person table, name Sam, aged 26.

UPDATE: update table set field=value where condition statement

e.g. update person set name='Lee' where id=10

Change the name of the record which id = 10 to Lee

DELETE: delete from table where condition statement

e.g. delete from person where id=10

Delete the record which id = 10 in table person.

In order to operate and manage SQLite database, Android system provides some related classes such as SQLiteOpenHelper, SQLiteDataBase, Cursor, you can check the other classes through the android.database.sqlite packages and android.database package in the Android develop documentation.

SQLiteOpenHelper is a helper class to manage database creation and update version management. You create a subclass to implement onCreate(SQLiteDatabase), onUpgrade (SQLiteDatabase, int, int) and optionally onOpen(SQLiteDatabase), of which this class takes care of opening the database if it exist, or creating it if it does not exist, and upgrading it as necessary. Transactions are used to make sure the database is always in a sensible state.

Main methods in SQLiteOpenHelper:

- **SQLiteDatabase getReadableDatabase()**: Create and/or open a database. This will be the same object returned by getWritableDatabase() unless some problem, for example a full disk requires the database to be opened for read-only. In this case, a read-only database object will be returned. If the problem is fixed, a future call to

getWritableDatabase() may succeed, in which the case of read-only database object will be closed and the read/write object will be returned in the future.

- **SQLiteDatabase getWritableDatabase()**: Create and/or open a database that will be used for reading and writing. The first time this is called, the database will be opened and onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) and/or onOpen(SQLiteDatabase) will be called. Once opened successfully, the database is catched, so you can call this method every time you need to write to the database. (Make sure to call close() when you no longer need the database.) Errors such as bad permissions or a full disk may cause this method to fail, but future attempts may succeed if the problem is fixed.
- **abstract void onCreate (SQLiteDatabase db)**: this method is called when the database is created for the first time, in which the creation of tables and the initial population of the tables should happen.
- **abstract void onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion)**: this method is called when the database needs to be upgraded. The implementation of this method is used to drop tables, add tables, or do anything else it needs to upgrade to the new schema version.
- **void onOpen (SQLiteDatabase db)**: this method is called when the database has been opened. The implementation should check isReadOnly() before updating the database.

When getting the instance of SQLiteDatabase by calling getReadableDatabase() or getWritableDatabase(), this class checking the database name to determine whether the database exist, creating it if it does not. Then callback onCreate() method, in the onCreate() method execute create table statement and add some initialization data the application need. If the database exist, system will check whether the database need to be upgraded without version. If the version is not the same as previous one, it need to update, and call onUpgrade() automatically. Execute the update of table structure and data in the method according to the business requirements.

SQLiteDatabase is a packaging SQLite database class provided by Android. It exposes methods to manage an SQLite database. SQLiteDatabase has methods to create, delete, execute SQL commands, and perform other common database management tasks. Database names must be unique within an application, not across all applications. Besides, there are two practical methods in SQLiteDataBase: execSQL() and rawQuery(). The execSQL() can execute SQL statement which can change the behavior such as insert, delete, update and create table. The rawQuery() is used to execute the select statement. Methods declaration:

- **execSQL(String sql,Object[] bindArgs)**: Execute a single SQL statement that will change the database, the result is a bool type.
- **execSQL(String sql)**: Execute a single SQL statement that is NOT a SELECT or any other SQL statement that returns data.
- **rawQuery(String sql,String[] selectionArgs)**: Runs the provided SQL and returns a

Cursor over the result set.

Cursor is a interface mainly used to store the query record. This interface provides random read-write access to the result set returned by a database query. Cursor implementations are not required to be synchronized so using a Cursor from multiple threads should perform its own synchronization when using the Cursor. Methods declaration:

- **move(int offset):** Move the cursor by a relative amount, forward or backward, from the current position. Positive offsets move forwards, negative offsets move backwards. If the final position is outside of the bounds of the result set then the resultant position will be pinned to -1 or count() depending on whether the value is off the front or end of the set, respectively.
- **moveToNext():** Move the cursor to the next row. This method will return false if the cursor is already past the last entry in the result set.
- **moveToPrevious():** Move the cursor to the previous row. This method will return false if the cursor is already before the first entry in the result set.
- **moveToFirst():** Move the cursor to the first row. This method will return false if the cursor is empty.
- **moveToLast():** Move the cursor to the last row. This method will return false if the cursor is empty.

Steps of manipulating database by class SQLiteDatabase:

(1) Create a database helper object, we use MyOpenHelper class here, specify the database name and version.

(2) Call method getReadableDatabase() or getWritableDatabase() to get SQLiteDatabase object which represents the connection to the database.

(3) Call the related methods of SQLiteDatabase object to execute add, delete, check, change operation.

(4) Manage the results of database operation such as check if it has inserted, deleted or updated successfully and turn the records into the form of a list.

(5) Close the database connection, recycling resources.

The essence of SQLite database is a file. When the database has been created successfully, the corresponding database file will generate in the folder / databases / of / data / data / package where the application lies in the mobile phone. The database can be viewed and exported by DDMS view. The database path in this case: data/data/iet.jxufe.cn.android.listviewdelayload/databases/news.db show as Figure12-2.



Figure 12-2 the figure of SQLite database storage path

After exporting the database, we can open database file through command sqlite3.exe under the directory tool of Android SDK like the command line window of MySQL. If you add directory tool to the environment variable, you can enter the database file directory through command line simply and type sqlite3 database name to open the database. If you have not added the directory to the environment variable, you need to enter the catalog of tool in Android SDK installation directory and type sqlite3 absolute path of database directory/database name to open the database and then execute the corresponding SQL statement.

Note: The Chinese contents of the database via the command line will be garbled in the command line.

12.4 Extension of Knowledge

In the process of developing applications with database operations, if there are some change in the method onCreate() in the database helper class, it seems like there are some changes in creating database statement or initializing. During the test period, it is necessary to make sure that the database in the phone has been deleted. Otherwise the system will not call onCreate() again since the database already exist on the phone and we cannot change the database.

When operating the database, usually we need to joint the SQL statement by the passing parameters. For example, the method used to query records in the database need a parameter named ID, then joint the SQL statement inside this method, execute query operation. If there are many parameters, it is rather inconvenient by jointing SQL statement and unsafe. A better solution is that represent the dynamic part through placeholder and "?" is the placeholder in SQL statement. Then assign these "?" respectively. The exeSQL() and rawQuery() in SQLitedatabase have provided SQL statement with placeholder. We have used a placeholder while initializing in this case. Such as "db.execSQL("insert into news_tb (image,title,info) values(?,?,?)", new String[]{R.drawable.news1+"", "Swiss confirm Ethiopian plane hijack", "An Ethiopian Airlines plane en route from Addis Ababa to Rome has been forced to land in Geneva after being hijacked, Swiss police say..."});".

12.5 Thinking and Exercises

(1) Describe the process of creating database. Will it go wrong if we query records directly when the database has not been created and execute create table statement in onCreate() while defining the helper class.

(2) Has the database extension any requirement?

(3) Any data can be inserted into any column. Developers don't need care about the data type (true/false).

(4) Which date type does not support by SQLite? (　)

A. BLOB　　B. INTEGER　　C. VARCHAR　　D. REAL

(5) Which description about SQLiteOpenHelper is incorrect? (　)

A. SQLiteOpenHelper is a database management tool class provides by Android, mainly used to create, open and update a database. It is a abstract class

B. To extend SQLiteOpenHelper, onCreate() method must be override

C. To extend SQLiteOpenHelper, onUpgrade() method must be override

D. To extend SQLiteOpenHelper, the construct method is not necessary

(6) To create a sub class of SQLiteOpenHelper, which method must be conclude in the new class? (　)

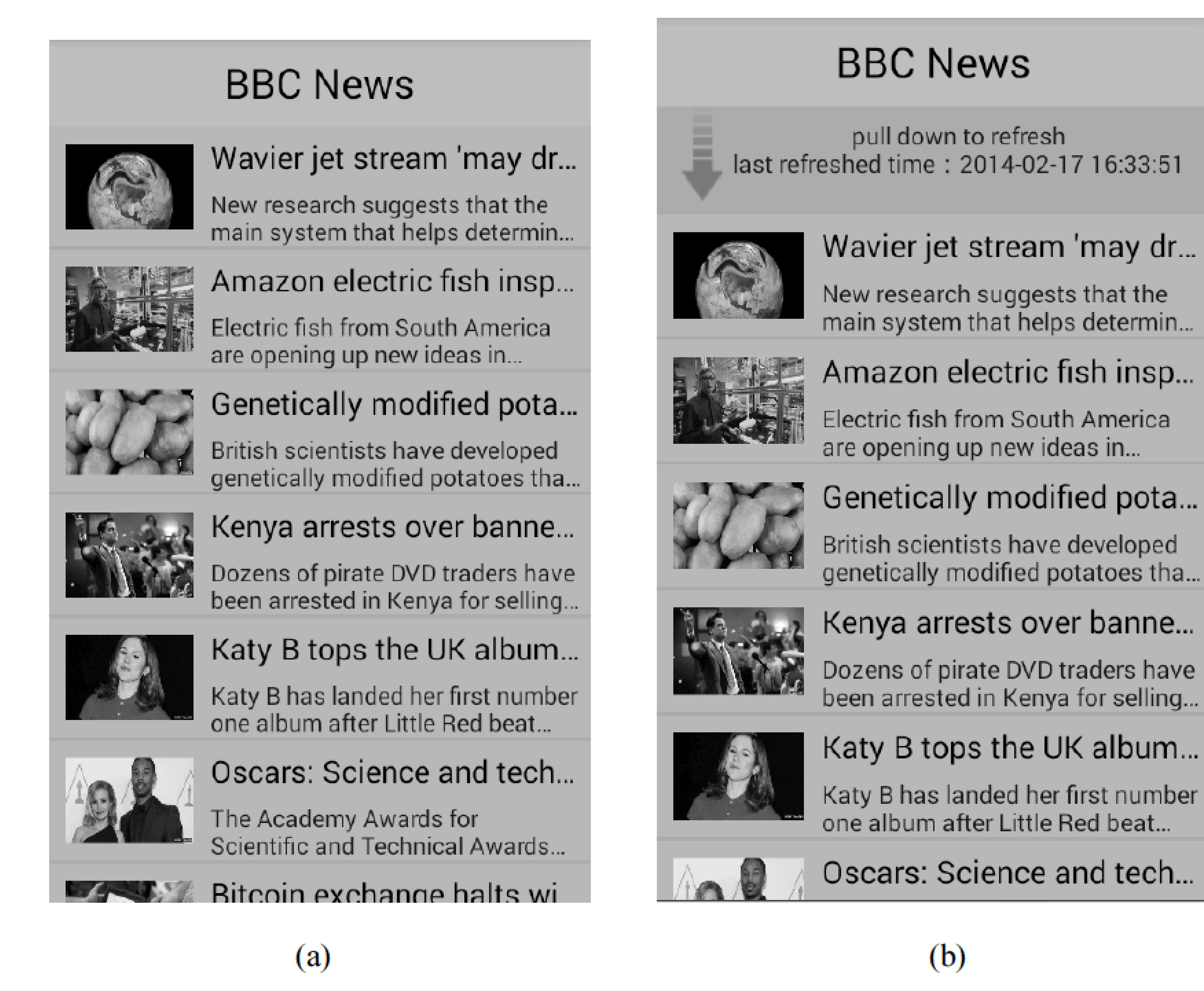
A. construct method　　B. onCreate()

C. onUpgrade()　　D. getReadableDatabase()

Chapter 13 BBC News—Drop Down Refresh ListView

13.1 Case Overview

This case focus on the effect of refreshing ListView by drop-down list. The contents of list are same as the previous case. Drop-down refresh often used in real-time updating data application like news client. When users read news, some new content may be released, but users often do not want to receive news alerts in real time. They want to refresh the page by themselves when they want to check the newest news. The effect of this case is user can pull down from the top of the list to refresh news, and the latest news is displayed at the top of the list. Our application running looks like Figure 13-1.



(a) (b)

Figure 13-1 the figure of the running results at different times



(c) (d) (e)

Figure 13-1(continued)

13.2 Key Code

Main interface layout file: 13\DropDownRefresh\res\layout\activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ccbbaa"
    android:orientation="vertical" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:drawableLeft="@drawable/logo"
        android:gravity="center_vertical"
        android:padding="10dp"
        android:text="@string/title"
        android:textColor="#000000"
        android:textSize="24sp" />
    <iet.jxufe.cn.android.dropdownrefresh.RefreshListView
        android:id="@+id/news"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#aabbcc"
```

```
        android:divider="#aaaaaa"
        android:dividerHeight="2dp"/>
</LinearLayout>
```

Layout file of each item in the news list: 13\DropDownRefresh\res\layout\item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/image"
        android:layout_width="75dp"
        android:layout_height="60dp"
        android:scaleType="fitXY"
        android:layout_margin="10dp"
        android:contentDescription="@string/imgInfo" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:paddingRight="5dp"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="#000000"
            android:singleLine="true"
            android:ellipsize="end"
            android:textSize="18sp" />
        <TextView
            android:id="@+id/info"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="left"
            android:textColor="#0000ee"
            android:textSize="14sp"
            android:maxLines="2"
            android:layout_marginTop="5dp"
            android:ellipsize="end"/>
    </LinearLayout>
</LinearLayout>
```

Layout file of refreshing news at the top: 13\DropDownRefresh\res\layout\refresh_header.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:background="#aaaabb"
    android:paddingBottom="5dp"
    android:orientation="horizontal" >
   <!—Progress bar used for showing loading, it is invisible by default.-->
    <ProgressBar
        android:id="@+id/progressbar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        style="?android:attr/progressBarStyle"/>
    <!—Show down arrow when drop down, and show upward arrow when release.-->
    <ImageView
        android:id="@+id/imageInfo"
        android:layout_width="24dp"
        android:layout_height="60dp"
        android:layout_marginRight="5dp"
        android:contentDescription="@string/imgInfo"/>
    <!—head message-->
    <TextView
        android:id="@+id/headerInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="#0000ff"
        android:textSize="14sp"
        android:paddingTop="10dp"
        android:paddingBottom="10dp"/>
</LinearLayout>
```

Program code: 13\DropDownRefresh\src\iet\jxufe\cn\android\dropdownrefresh\MainActivity.java

```java
public class MainActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        //Hide the title bar
        setContentView(R.layout.activity_main);
    }
}
```

Database helper class: 13\DropDownRefresh\src\iet\jxufe\cn\android\dropdownrefresh\MyOpenHelper.java

```
public class MyOpenHelper extends SQLiteOpenHelper {
    public String createTableSQL="create table if not exists news_tb" +
            "(_id integer primary key autoincrement,image,title,info)";
            //Create table statement
    //Constructor method
    public MyOpenHelper(Context context, String name, CursorFactory
            factory,
            int version) {
        super(context, name, factory, version);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(createTableSQL);
        //Initialization insert statement is omitted here
    }
    //the method invoked when the version of database is updated
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
        newVersion) {
    System.out.println("version changed:"+oldVersion+"-->"+ newVersion);
    }
}
```

Custom list class: 13\DropDownRefresh\src\iet\jxufe\cn\android\dropdownrefresh\MyOpenHelper.java

```
public class RefreshListView extends ListView {
    private SimpleAdapter simpleAdapter;
    //Adapter associated with data source and ListView
    //collection used to store news data have loaded
    private LinkedList<Map<String, Object>> newsList;
    private LinearLayout mHeaderView = null;//head layout
    private TextView headerInfo;
     //view used to display text information at the top.
    private ImageView imageInfo;//view used to display image information
                                   //at the top
    private ProgressBar mProgressBar;//progress bar at the top
    private int mHeaderHeight;//the height of head view
    //private boolean hasNew;//flag of whether has new records
    private int mCurrentScrollState;//the state of current scroll bar
    private MyOpenHelper mHelper;
    private SQLiteDatabase mDB;
    //custom states during the pull down refresh process
    private final static int NONE_PULL_REFRESH = 0;//normal state
    private final static int ENTER_PULL_REFRESH = 1;//pull-down refresh
                                                      //state
```

```
    private final static int OVER_PULL_REFRESH = 2;//release to refresh state
    private final static int EXIT_PULL_REFRESH = 3;//release to load state
    private int mPullRefreshState = 0; //record refresh state, the
                                       //default is the normal state
    private float mDownY;//the coordinate when press down
    private float mMoveY;//the coordinate after movement
    private SimpleDateFormat mSimpleDateFormat = new SimpleDateFormat(
            "yyyy-MM-dd hh:mm:ss");//the format of time display
    //some message signs when handling messages by Handler
    private final static int REFRESH_BACKING = 0;//rebounding
    private final static int REFRESH_BACKED = 1; //need to refresh after
                                                 //rebound
    private final static int REFRESH_RETURN = 2;//do not need to refresh
                                                //after rebound
    private final static int REFRESH_DONE = 3;//data loading is finish
    public RefreshListView(Context context) {
        this(context, null);
    }
    public RefreshListView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context);//initialize operation
        setOnScrollListener(new MyScrollListener());//add scroll event
                                                    //listener
        setSelection(1);//set the selected item, the default is 0, head
                        //View is also considered as an item
    }
    public void init(final Context context) {
        mHelper = new MyOpenHelper(context, "news.db", null, 1);
        mDB = mHelper.getReadableDatabase();
        LayoutInflater layoutInflater = (LayoutInflater) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        //get the LayoutInflater, convert the layout file into a View object
        mHeaderView = (LinearLayout) layoutInflater.inflate(
                R.layout.refresh_header, null);
                //convert the layout file into a View object
        addHeaderView(mHeaderView);//add the head View to the list.
        newsList = getData("select * from news_tb order by _id desc limit
0,6",
                null);//get initial information
        simpleAdapter = new SimpleAdapter(context, newsList, R.layout.item,
                new String[] { "image", "title", "info" }, new int[] {
                        R.id.image, R.id.name, R.id.info });
        this.setAdapter(simpleAdapter);
        //get the corresponding widgets in the layout according to id.
```

```
        HeaderInfo = (TextView)mHeaderView.findViewById(R.id. headerInfo);
        imageInfo = (ImageView) mHeaderView.findViewById(R.id.imageInfo);
        mProgressBar = (ProgressBar) mHeaderView.findViewById
         (R.id.progressbar);
        reset();//initial state
        measureView(mHeaderView);//Specify the size of the head view
        mHeaderHeight = mHeaderView.getMeasuredHeight();
         //get the height of the head View
    }
    private void measureView(View child) {
        ViewGroup.LayoutParams p = child.getLayoutParams();
         //Get the layout parameter of widget
        if (p == null) {
            p = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.
                    MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT);
        }
        int childWidthSpec = ViewGroup.getChildMeasureSpec(0, 0,
         p.width);      //get width requirements
        int lpHeight = p.height;
        int childHeightSpec;
        if (lpHeight > 0) {
            childHeightSpec = MeasureSpec.makeMeasureSpec(lpHeight,
                    MeasureSpec.EXACTLY);
           //the height of the child widget specified by the parent
           //container, it is a precise value without the consideration
           //to the size of the content of child widget
        } else {
            childHeightSpec = MeasureSpec.makeMeasureSpec(0,
                    MeasureSpec.UNSPECIFIED);
            //the height of chile widget is unlimited, it can be any value.
        }
        child.measure(childWidthSpec,  childHeightSpec);//width  and height of the child widget
    }
    //Get news data according to the query statement
    public LinkedList<Map<String, Object>> getData(String sql, String[] args) {
        LinkedList<Map<String,Object>>  list  =  new  LinkedList<Map<String, Object>>();
        Cursor cursor = mDB.rawQuery(sql, args);//query the qualified record
        while (cursor.moveToNext()) {
        //iterate each record, get the appropriate data and save the data
        //to the collection
            Map<String, Object> item = new HashMap<String, Object>();
```

```
                item.put("image", cursor.getString(cursor.getColumnIndex
                        ("image")));
                item.put("title", cursor.getString(cursor.getColumnIndex
                        ("title")));
                item.put("info", cursor.getString(cursor.getColumnIndex
                        ("info")));
                list.add(item);
            }
            return list;
        }
        public class MyScrollListener implements OnScrollListener {
        //scroll event listener
            public void onScrollStateChanged(AbsListView view, int scrollState){
                mCurrentScrollState = scrollState;
                //record the current state of the scroll bar
            }
            public void onScroll(AbsListView view, int firstVisibleItem,
                    int visibleItemCount, int totalItemCount) {
                    //called when scrolling
                if (mCurrentScrollState == SCROLL_STATE_TOUCH_SCROLL
                        && firstVisibleItem == 0
                        && (mHeaderView.getBottom() >= 0 &&
                        mHeaderView.getBottom() < mHeaderHeight)) {
//when your finger is sliding on the scroll bar and the head View is not
//fully displayed, entered into the pull-down refresh state
                    if (mPullRefreshState == NONE_PULL_REFRESH) {
                        mPullRefreshState = ENTER_PULL_REFRESH;
                    }
                } else if (mCurrentScrollState == SCROLL_STATE_TOUCH_SCROLL
                        && firstVisibleItem == 0
                        && (mHeaderView.getBottom() >= mHeaderHeight)) {
                //enter and go to refresh state when scroll to or exceed the
                //height of the head View
                    if (mPullRefreshState == ENTER_PULL_REFRESH
                            || mPullRefreshState == NONE_PULL_REFRESH) {
                        mPullRefreshState = OVER_PULL_REFRESH;
                        //a display change when entering into and go to
                        //refresh state
                        mDownY = mMoveY;//record the coordinate when
                                //entering and go to refresh state
                        headerInfo.setText("leave off will refresh\nlast
                                refreshed time: "
                                + mSimpleDateFormat.format(new Date()));
                        imageInfo.setImageResource(R.drawable.up_arrow);
                    }
```

```
            } else if (mCurrentScrollState == SCROLL_STATE_TOUCH_SCROLL
                    && firstVisibleItem != 0) {
                //when the visible item is not the first item, it means
                //that the application has not been dragged to the top yet
                if (mPullRefreshState == ENTER_PULL_REFRESH) {
                    mPullRefreshState = NONE_PULL_REFRESH;
                }
            } else if (mCurrentScrollState == SCROLL_STATE_FLING
                    && firstVisibleItem == 0) {
                //fly-slide state, can not show header and can not affect
                //the normal flying slider
                //the position can be recovered only under normal
                //circumstances
                if (mPullRefreshState == NONE_PULL_REFRESH) {
                    setSelection(1);
                }
            }
        }
    }
    public boolean onTouchEvent(MotionEvent ev) {
        switch (ev.getAction()) {
        case MotionEvent.ACTION_DOWN:
            mDownY = ev.getY();
            break;
        case MotionEvent.ACTION_MOVE:
            mMoveY = ev.getY();
            if (mPullRefreshState == OVER_PULL_REFRESH) {
                //The following mDownY has been changed in the second
                //else block in the method onScroll
                mHeaderView.setPadding(0, (int)((mMoveY- mDownY)/2), 0,
                mHeaderView.getPaddingBottom());
                //the top margin is 1/2 of drag distance.
            }
            break;
        case MotionEvent.ACTION_UP:
            if (mPullRefreshState == OVER_PULL_REFRESH
                    || mPullRefreshState == ENTER_PULL_REFRESH) {
                final Timer timer=new Timer();
                timer.schedule(new TimerTask() {
                    public void run() {
                        while(mHeaderView.getPaddingTop()>1){
                            Message msg=mHandler.obtainMessage();
                            msg.what = REFRESH_BACKING;
                            //send bounce message
```

```
                        mHandler.sendMessage(msg);
                    }
                    Message message=mHandler.obtainMessage();
                    if (mPullRefreshState == OVER_PULL_REFRESH) {
                        //if the original state is released
                        message.what = REFRESH_BACKED;
                        //complete bounce, loading data
                    } else {
                        message.what = REFRESH_RETURN;
                        //complete bounce, do not load data
                    }
                    mHandler.sendMessage(message);
                    timer.cancel();
                }
            }, 0,100);
        };
        break;
    }
    return super.onTouchEvent(ev);
}
private Handler mHandler = new Handler() {//create a handler object
                //to send, receive and handle messages
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case REFRESH_BACKING://handle rebound, make the top margin
                //becomes 3/4 of previous
            mHeaderView.setPadding(0, (int)
            (mHeaderView.getPaddingTop()*0.75f),
            0,mHeaderView.getPaddingBottom());
            break;
        case REFRESH_BACKED://load data
            imageInfo.setVisibility(View.GONE);
            mProgressBar.setVisibility(View.VISIBLE);
            headerInfo.setText("refreshing...\nlast refreshed time: "
                    + mSimpleDateFormat.format(new Date()));
            mHandler.postDelayed(new Runnable() {
                public void run() {
                    refreshing();
                }
            }, 3000);
            break;
        case REFRESH_RETURN://when the rebound is over and do not
                //need to load the data, restitution
        case REFRESH_DONE://recover the default original state after
                //refreshing
```

```
                    reset();
                    break;
                default:
                    break;
                }
            }
        };
        public void reset(){//recover to original state
            headerInfo.setText("pull down to refresh\nlast refreshed time: "
                    + mSimpleDateFormat.format(new Date()));
            mProgressBar.setVisibility(View.GONE);
            imageInfo.setVisibility(View.VISIBLE);
            imageInfo.setImageResource(R.drawable.down_arrow);
            mHeaderView.setPadding(0,0,0,mHeaderView.
                                        getPaddingBottom());
            mPullRefreshState = NONE_PULL_REFRESH;
            setSelection(1);
        }
        public void refreshing() {
            LinkedList<Map<String, Object>> list = getData(
                    "select * from news_tb order by _id desc limit ?,?",
                    new String[] { simpleAdapter.getCount() + "", 2 + "" });
            if (list != null && list.size() != 0) {
                for (int j = 0; j < list.size(); j++) {
                    newsList.addFirst(list.get(j));
                }
                simpleAdapter.notifyDataSetChanged();//updated list display
                mPullRefreshState = EXIT_PULL_REFRESH;
                //changes state, sending a message
                Message message = mHandler.obtainMessage();
                message.what = REFRESH_DONE;
                mHandler.sendMessage(message);
            }
        }
    }
```

13.3 Code Analysis

Drop-down refresh is always used in updating application data in the real time. Every reloading, the application will send a request to the server to load the new generated records since last refresh like SINA micro-blog client and 163 news client. The principle of drop-down refresh is adding an item used to display the information of the user to pull down the list at the

top of the ListView. By default, this item is hidden which means the list begins to show from the second item. When the user pulls down the scroll bar, the user will be prompted how to handle according to whether the first item has been fully displayed. The prompt messages include: pull down to refresh, let go to refresh, refreshing and so on. 4 states have been involved in this process.

- **NONE_PULL_REFRESH**: Common state. This is the default state and we cannot see the first item under this state.
- **ENTER_PULL_REFRESH**: Enter into refreshing state. The first item will prompt the user pull down to refresh the information, and display the down arrow at the same time under this state. The state lasts for the time of scroll bar's sliding, from the first sight of the bottom of first item to the fully displayed.
- **OVER_PULL_REFRESH**: Enter into letting go refresh state. The first item will prompt the user let go to refresh the information, and display the upward arrow at the same time under this state. The state lasts for the time when the first item fully displayed, the scroll bar continue sliding until the user let go.
- **EXIT_PULL_REFRESH**: Let go rebound state. The first item will prompt user information is refreshing and show refresh progress bar at the same time under this state. The user enters this state when letting go.

The four states above switch are designed according to the change of ListView scrolling state. Therefore, we need to add a scroll event listener for the ListView widget and record the coordinates of scroll bar where the user pressed and monitor the event of user release. So we also need to add a touch event listener for the ListView. The scroll event listener and touch event listener codes, from 90th-169th line. The main processes of pull-down refresh (see Figure 13-2):

(1) Pull down, from nothing to show part HeaderView at the top of ListView, prompts users that pull down can refresh. The refresh state turns to ENTER_PULL_REFRESH from NONE_PULL_REFRESH.

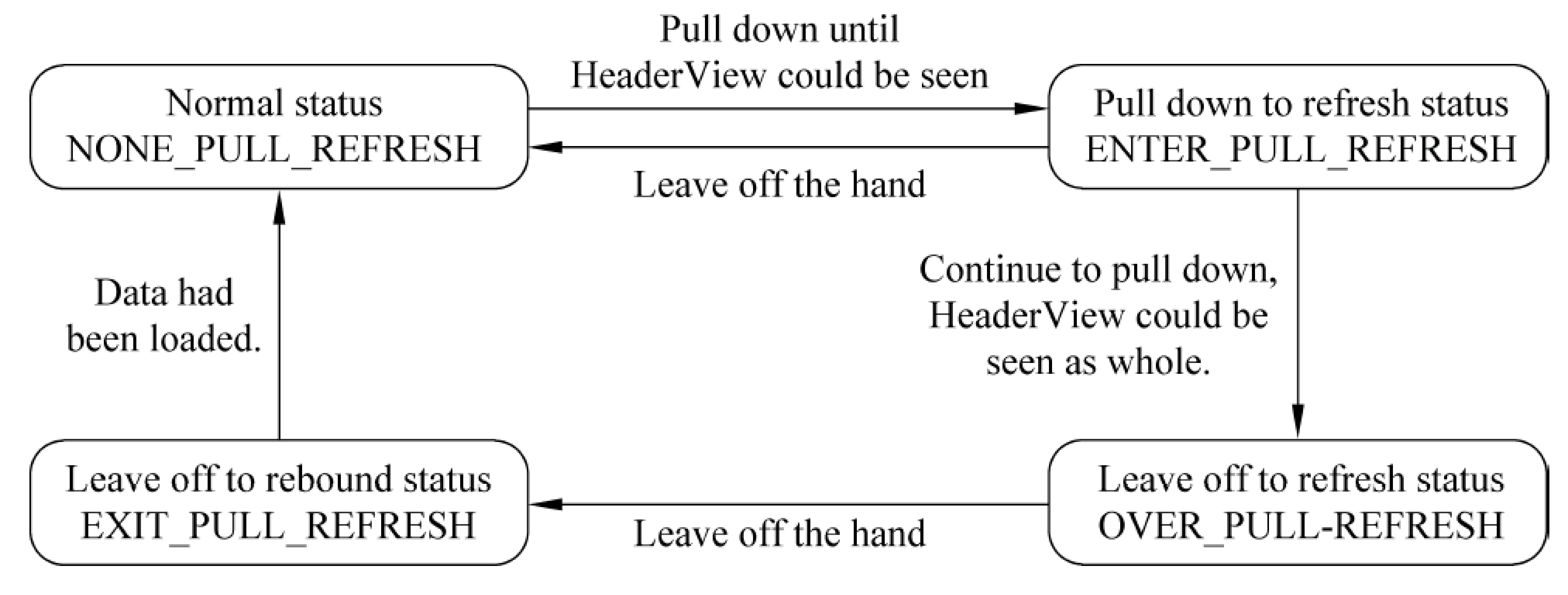


Figure 13-2 flow chart for drop-down refresh

(2) Keep pulling down, from showing part of HeaderView to fully displayed, reach the minimum height requirement to refresh and prompts users that let go to refresh. But still allows

users pulling down by increasing margins of HeaderView. The refresh state turns to OVER_PULL_REFRESH from ENTER_PULL_REFRESH.

(3) The user lets go. The user may drop down far more than the height of HeaderView, so it should bounce back to show the head tip control only by decreasing margins constantly. Then prompts the user that the application is refreshing. The status turns to NONE_PULL_REFRESH from EXIT_PULL_REFRESH.

13.4 Extension of Knowledge

In the previous case delay loading, news data is stored in the ArrayList collection, the new records by delay loading will add to the end of list. But in this case the new acquired data will be added at the beginning of the list and it is hard to make it come true by ArrayList collection, so we use LinkedList instead. Both ArrayList and LinkedList are implementation class of interface List. The basement of ArrayList is based on the data structure of dynamic arrays, while LinkedList is based on the data structure of list. They both have advantages and disadvantages in terms of performance, the developers can choose to use depending on the circumstances. Specific features:

(1) The cost of adding an element at the end of the list is constant to both ArrayList and LinkedList. For the ArrayList, add an item inside the array point to the added element and it may lead to re-allocate the array occasionally. For the LinkedList, the cost is uniform, distribute an internal Entry object.

(2) Insert or delete an element in the ArrayList means that the left elements in the list will be moved together ; but the cost of insert or delete an element in the LinkedList is uniform .

(3) LinkedList does not support efficient random element access.

(4) The space waste of ArrayList is mainly reflected in the space reserved for a certain capacity at the end of the list. For LinkedList, it is reflected in that every element needs to consume considerable space.

Generally, if the operation is to add a new item in the end of list instead of the front or middle, and need to random access to the elements inside it, ArrayList is our first choice. If the operation is to add or delete data in the front or middle of the list and access the elements in order, we choose LinkedList.

Class LinkedList has offered us method addFirst() and addLast(), we can add new elements at the beginning or the end of the list easily. This meet the requirement of this case, so we choose LinkedList here.

13.5 Thinking and Exercises

Please describe the pull-down refresh process and the conversion relationship between several states.

Chapter 14 ExpandableListView Widget

14.1 Case Overview

This case mainly introduced the usage of ExpandableListView widget. In this case, the province is a list, when you unfold an item of the province list, it will display the city list. The key point of this case is how to make the city list associated with its own provinces. Just like the ListView widget, the ExpandableListView cannot be associated with data through itself, it need help of relevant Adapter. You can define the Adapter by yourself or use the Adapter class provided by system. The data is very easy; we use the TextView widgets to show words. In the following case, it adopted SimpleExpandableListAdapter provided by system. When program running, it looks like Figure 14-1.

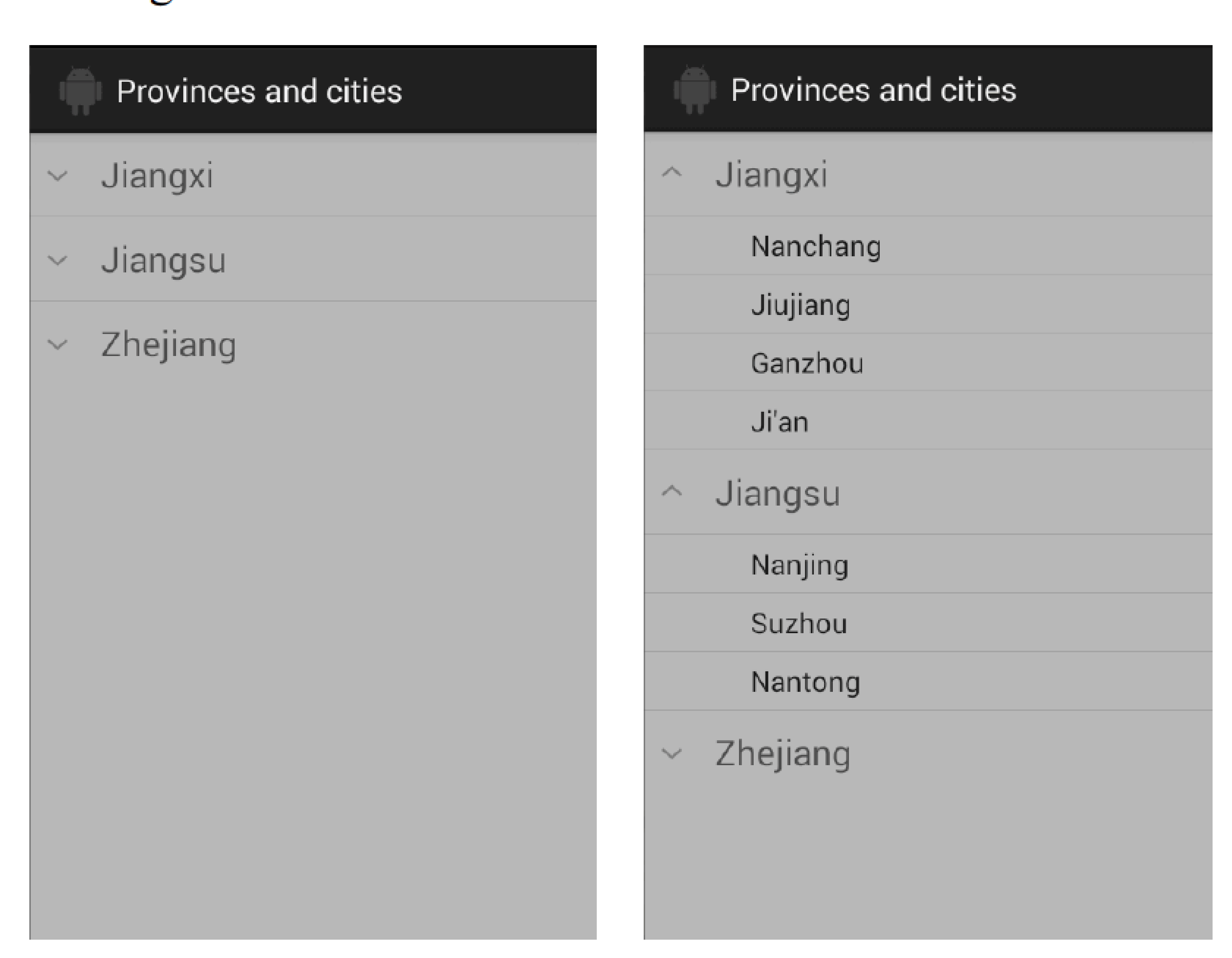


Figure 14-1 the figure of the running results at different status

14.2 Key Code

Main interface layout file: 14\ ProvinceAndCityList \res\layout\activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc">
    <ExpandableListView
        android:id="@+id/mExpandableListView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

Display province layout file: 14\ ProvinceAndCityList \res\layout\province.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical">
    <TextView
        android:id="@+id/group"
        android:paddingLeft="40dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#ff0000"
        android:paddingTop="10dp"
        android:paddingBottom="10dp"
        android:textSize="20sp" />
</LinearLayout>
```

Display city layout file: 14\ ProvinceAndCityList \res\layout\city.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical">
    <TextView
        android:id="@+id/child"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#0000ff"
        android:paddingLeft="60dp"
        android:paddingTop="5dp"
        android:paddingBottom="5dp"
        android:textSize="16sp" />
</LinearLayout>
```

14\ ProvinceAndCityList\src\iet\jxufe\cn\android\provinceandcitylist\MainActivity.java

```
public class MainActivity extends Activity{
    private ExpandableListView mExpandableListView;
    private String[] provinces = new String[] {"Jiangxi", ""Jiangsu",
"Zhejiang" };//provinces information
    private String[][] cities = new String[][] { {"Nanchang", "Jiujiang",
"Ganzhou", "Ji’an" },
            {"Nanjing", "Suzhou", "Nantong"}, { "Hangzhou", "Jinhua" } };
            //cities information
    private List<Map<String, String>> provinceItems = new
     ArrayList<Map<String, String>>();
    //collection to save the information of all provinces
    private List<List<Map<String, String>>> cityItems = new
   ArrayList<List<Map<String,String>>>();
   //collection to save the information of all cities
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        This.setContentView(R.layout.activity_main);
    mExpandableListView=(ExpandableListView)findViewById
                                (R.id.mExpandableListView);
        init();//execute an initialization operation
        SimpleExpandableListAdapter adapter = new
         SimpleExpandableListAdapter(
                this,provinceItems,R.layout.province,new String[]{"group"},
                new int[] { R.id.province }, cityItems, R.layout.city,
                new String[] { "child" }, new int[] { R.id.city });
                //use the adapter provided by system
        mExpandableListView.setAdapter(adapter);//associated the data
    }
    public void init() {//execute the initialization operation
        for (int I = 0; I < provinces.length; I++) {
            //iterate the provinces array, associating every province
            //with the cities
            Map<String, String> provinceItem = new HashMap<String, String>();
            provinceItem.put("group", provinces[i]);
            provinceItems.add(provinceItem);
            //the collection to save the cities in the province
            List<Map<String,String>> cityList=new
            ArrayList<Map<String,String>>();
            for (int j = 0; j < cities[i].length; j++) {
                Map<String, String> cityItem = new HashMap<String,
               String>();
                cityItem.put("child", cities[i][j]);
                cityList.add(cityItem);
```

```
                }
                //collection to save cities in all the provinces
                cityItems.add(cityList);
            }
        }
}
```

14.3 Code Analysis

This case mainly introduced a special list——ExpandableListView. We can expand each item in the ExpandableListView, then we get a new list. It is often used in practical applications, such as, each province has their cities, each product has its own subproduct, each book has its own category, and each chapter has its sections.

Like the ListView, we can store the information of each item in a set. For example, we store the information of province in a set, and store the information of city in another set. However, there are some relationships in the cities whether they belong to the same province or not. So, we need to classify them according to the relationships. Above all, we know the set which storing city information is a special collection, the elements in the city set mean all these city belongs to one province, and this set is also a collection, the statement of city set as followed :**List<List<Map<String, String>>>** cityItems.

Now we have the data source, and we need to combine the data with the list widget through the SimpleExpandableListAdapter provided by system. When the object was created, it need to pass 9 parameters, the constructor of the class as followed: SimpleExpandableListAdapter (Context context, List<? extends Map<String, ?>> group Data, int group Layout, String[] groupFrom, int[] groupTo, List<? extends List<? extends Map<String, ?>>> childData, int childLayout, String[] childFrom, int[] childTo)，9 parameters are:

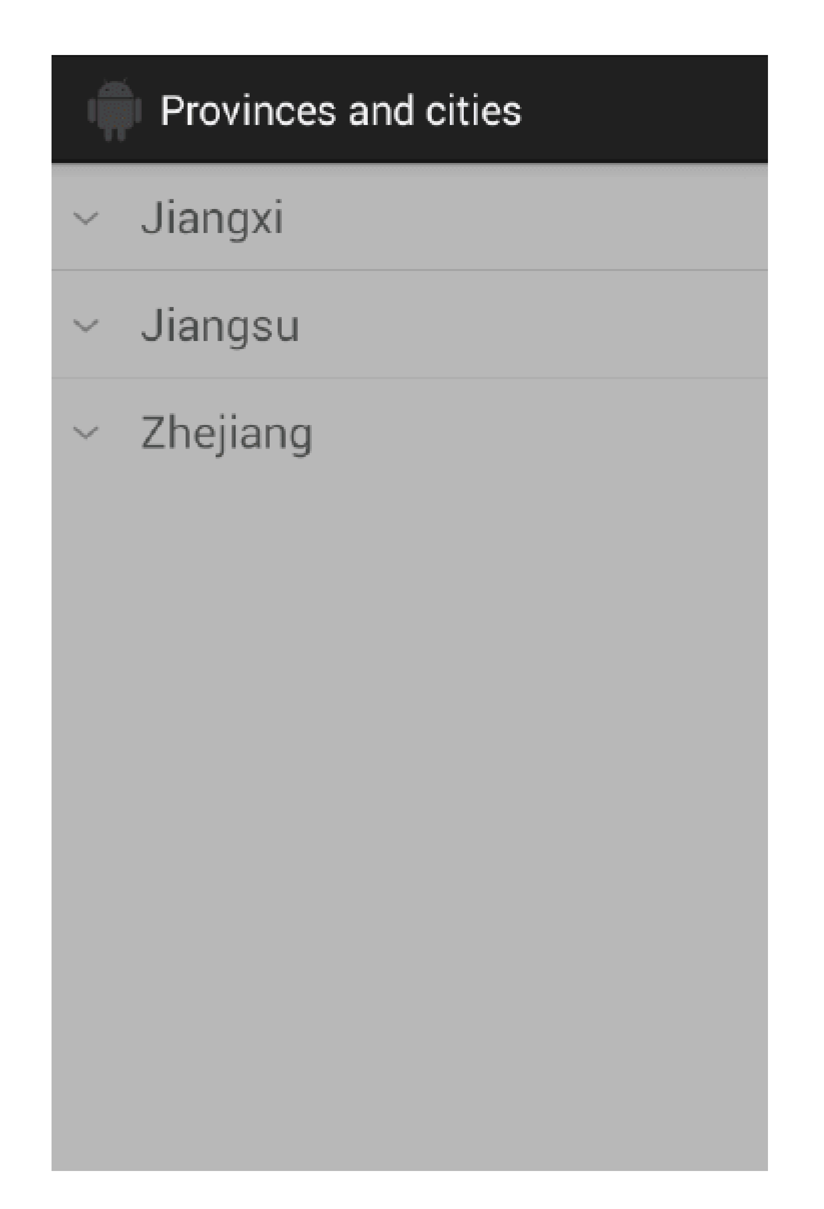
- **context**: The context where the ExpandableListView associated with this SimpleExpandableListAdapter is running.
- **group Data**: A List of Maps. Each entry in the List corresponds to one group in the list. The Maps contain the data for each group, and should include all the entries specified in "groupFrom".
- **groupLayout**: Resource identifier of a view layout that defines the views for a group. The layout file should include at least those named views defined in "group To".
- **groupFrom**: A list of keys that will be fetched from the Map associated with each group.
- **groupTo**: The group views that should display column in the "group From" parameter. These should all be TextViews. The first N views in this list are given the values of the first N columns in the groupFrom parameter.

- **childData**: A List of List of Maps. Each entry in the outer List corresponds to a group (index by group position), each entry in the inner List corresponds to a child within the group (index by child position), and the Map corresponds to the data for a child (index by values in the child From array). The Map contains the data for each child, and should include all the entries specified in "childFrom".
- **childLayout**: Resource identifier of a view layout that defines the views for a child. The layout file should include at least those named views defined in "childTo".
- **childFrom**: A list of keys that will be fetched from the Map associated with each child.
- **childTo**: The child views that should display column in the "childFrom" parameter. These should all be Text Views. The first N views in this list are given the values of the first N columns in the child From parameter.

We can use the SimpleExpandableListAdapter to realize the expandable effect, however the function is very limited and the list item can only be text. If we want to display a more complex list like an item containing a picture, we need to use the custom Adapter.

14.4 Extension of Knowledge

Similar to ListView, we can use some common Adapters to create our own Adapter, and we can also define Adapter by inheriting the base Adapter class provided by the system. For the extended drop-down list, the base Adapter class is the BaseExpandableListAdapter(). When we use the custom Adapter, we need to override the relevant method in the BaseExpandableListAdapter class. Although there will be a lot of codes, but more flexible and less limit. The case showed below adopted a custom Adapter to realize a complex list that we added an ImageView widget to the left of TextView widget, which displayed the city name, our application looks like Figure 14-2 when running.



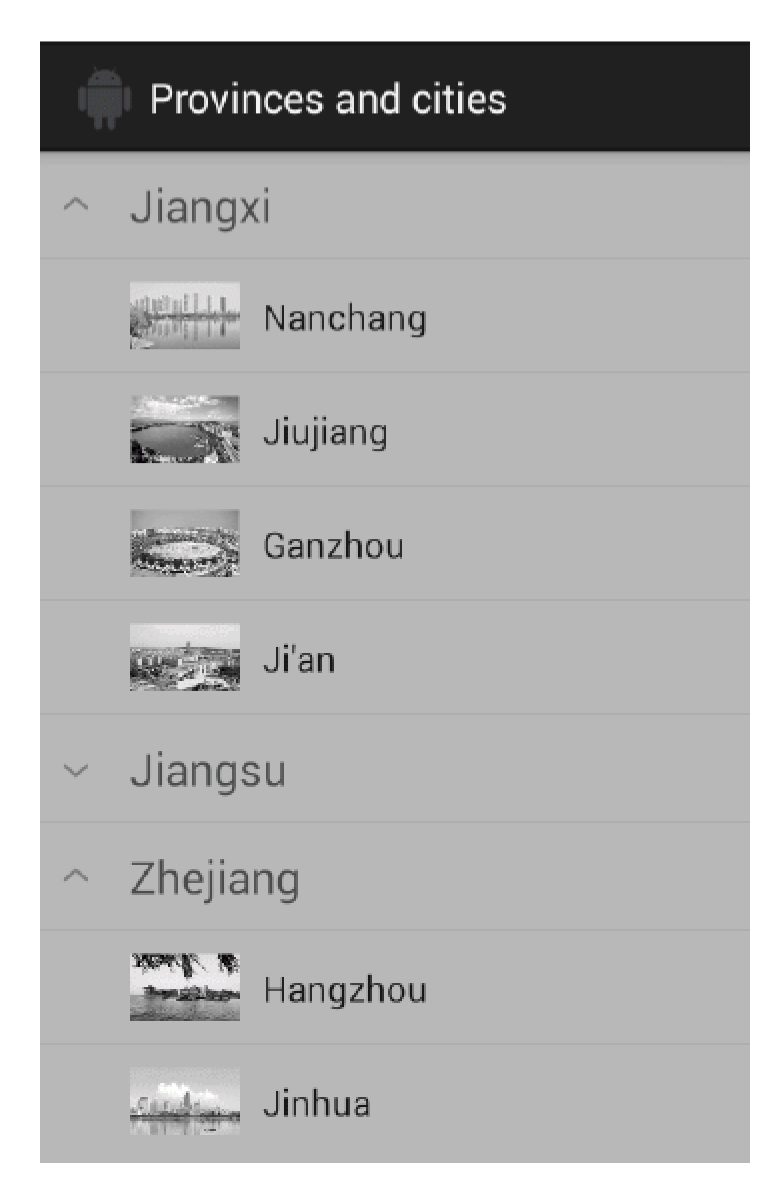


Figure 14-2　the figure of the running results at different status

The main layout codes and the detail layout codes about the city information is alike, so not listed here.

Display city layout file: 14\ ProvinceAndCityList\res\layout\city.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:gravity="center_vertical">
    <ImageView
        android:id="@+id/cityImg"
        android:layout_width="50dp"
        android:layout_height="30dp"
        android:layout_marginTop="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginBottom="10dp"
        android:layout_marginLeft="40dp"
        android:contentDescription="@string/imgInfo"/>
    <TextView
        android:id="@+id/city"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#0000ff"
        android:paddingTop="5dp"
        android:paddingBottom="5dp"
        android:textSize="16sp" />
</LinearLayout>
```

The main program:
14\ProvinceAndCityDIY\src\iet\jxufe\cn\android\provinceandcitydiy\MainActivity.java

```
public class MainActivity extends Activity {
    private ExpandableListView mExpandableListView;
    private String[] provinces = new String[] { "Jiangxi", "Jiangsu", "Zhejiang" };
    private String[][] cities = new String[][] { { "Nanchang", "Jiujiang", "Ganzhou",
                "Ji'an"},{"Nanjing","Suzhou","Nantong"},{"Hangzhou", "Jinhua"}};
    private int[][] cityImgIds = new int[][] {
                {R.drawable.nanchang, R.drawable.jiujiang, R.drawable.ganzhou,
                R.drawable.jian },{R.drawable.nanjing, R.drawable.suzhou,
                R.drawable.nantong},{R.drawable.hangzhou, R.drawable.jinhua}};
                //array to save ids of image of the city
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.activity_main);
        mExpandableListView = (ExpandableListView) findViewById
                                   (R.id.mExpandableListView);
        mExpandableListView.setAdapter(new MyAdapter());
    }
    private class MyAdapter extends BaseExpandableListAdapter{
    //custom Adapter class
        public int getGroupCount() {// get the count of group
            return provinces.length;
        }
        public int getChildrenCount(int groupPosition) {
        //get the count of children in the group
            return cities[groupPosition].length;
        }
        public Object getGroup(int groupPosition) {
        //get the group in groupPosition
            return null;
        }
        public Object getChild(int groupPosition, int childPosition) {
            //get the specified child in childPosition in groupPosition
            return null;
        }
        public long getGroupId(int groupPosition) {
            //get group id in groupPosition
            return 0;
        }
        public long getChildId(int groupPosition, int childPosition) {
            //get the child id in childPosition in groupPosition
            return 0;
        }
        public boolean hasStableIds() {//whether has stable id
            return false;
        }
        public View getGroupView(int groupPosition, boolean isExpanded,
                View convertView, ViewGroup parent) {
                  //get the view in groupPosition
            View groupView=getLayoutInflater().inflate(R.layout.province,
              null);
            TextView provinceText = (TextView) groupView.findViewById
                (R.id.province);
            provinceText.setText(provinces[groupPosition]);
            return groupView;
        }
        public View getChildView(int groupPosition, int childPosition,
```

```
                boolean isLastChild, View convertView, ViewGroup parent) {
                //get the view in childPosition in groupPosition
                View childView=getLayoutInflater().inflate(R.layout.city,
null);
                TextView cityText = (TextView) childView.findViewById
                (R.id.city);
                ImageView cityImg = (ImageView) childView.findViewById
                (R.id.cityImg);
                cityText.setText(cities[groupPosition][childPosition]);
                cityImg.setImageResource(cityImgIds[groupPosition]
                                          [childPosition]);
                return childView;
            }
            public boolean isChildSelectable(int groupPosition, int childPosition) {
                //whether the sub item is selectable
                return false;
            }
        }
}
```

You need to override 10 abstract methods in the class inheriting the BaseExpandableListAdapter to implement custom adapter, the four important methods are getGroupCount(), getChildrenCount(), getGroupView() and getChildView(). According to the four methods, we can obtain the number of groups, the number of children items in a group, the display view of each group and the display view of children item is in each group. You can selectively override other methods according to your requirement, otherwise use the default one. For example, the isChildSelectable() method means whether the child is selectable, if you need to add the selected event processing on the child, the method must return true, otherwise it cannot perform the selected event processing.

14.5 Thinking and Practice

In this case, the relevant information of provinces and cities specified temporary by the procedures, try to establish a database, save the related information, and then through the classification search, gain the related information, then display the information in the extensive list.

Chapter 15　Product Category —Custom Multi-level List

15.1　Case Overview

This case mainly realizes the custom multilevel list. In practical applications, there are many multilevel systems like object classification, blood inheritance of human beings and so on. But Android only provides us two widgets: the primary list ListView and the secondary list ExpandableListView which far failed to meet our requirement. So we need to use our knowledge to design a similar effect. This case is still using ListView essentially with a custom Adapter.

When you click an item, the system will judge whether there is a child item belongs to it. If it exists, the system should judge whether it has spread, if not, spread it. And if it has already spread, close it. Our application looks like Figure 15-1 when running.

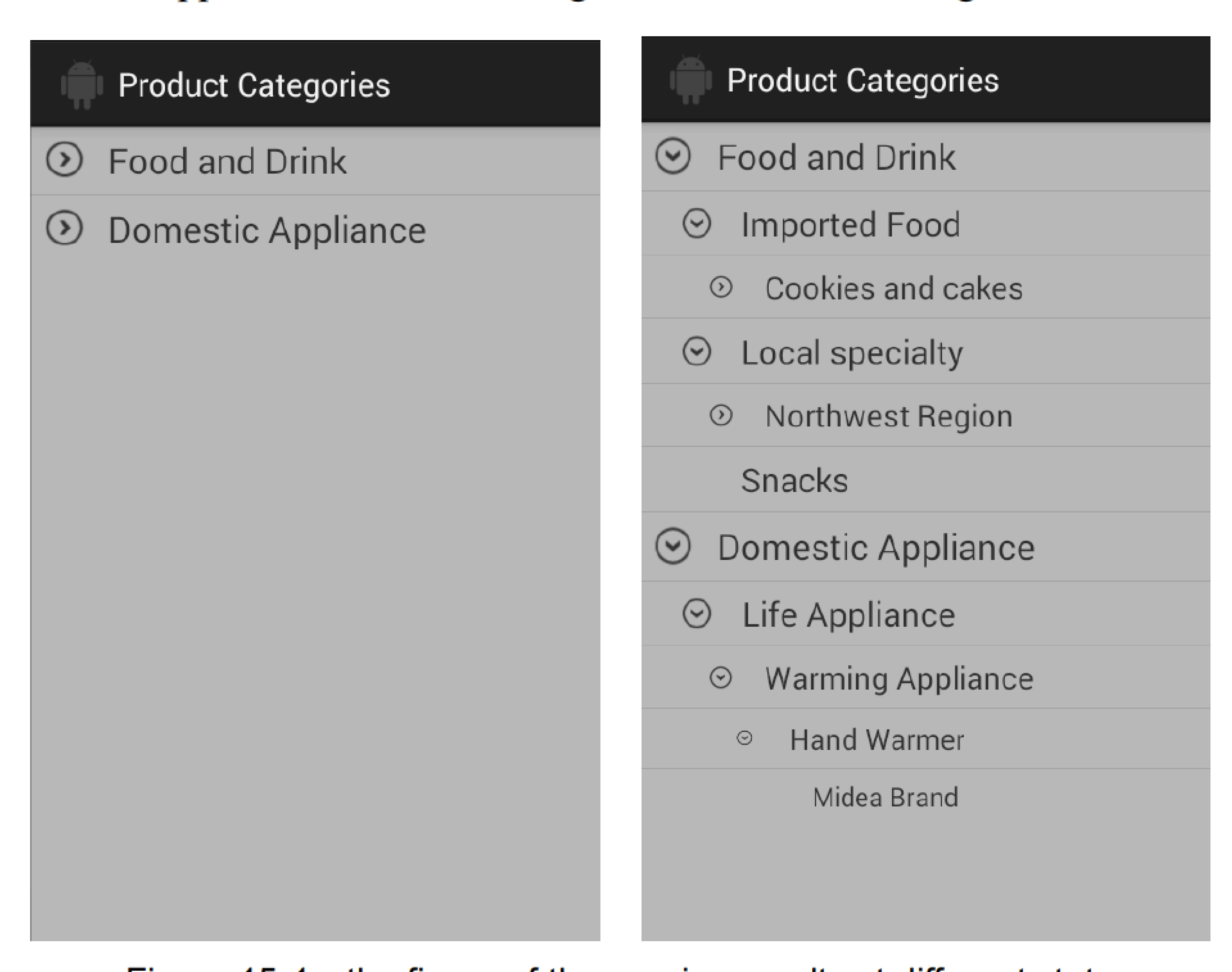


Figure 15-1　the figure of the running results at different status

15.2 Key Code

Main interface layout file: 15\ ProductCatagories \res\layout\activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc">
    <ListView
        android:id="@+id/mListView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</RelativeLayout>
```

Layout file of each item in the list: 15\ ProductCatagories \res\layout\treeview_item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="40dp"
        android:contentDescription="@string/imgInfo"/>
    <TextView
        android:id="@+id/text"
        android:textSize="20sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:gravity="center"/>
</LinearLayout>
```

Custom list item: 15\ ProductCatagories\src\iet\jxufe\cn\android\productcatagories\Element.java

```
public class Element {
    private String text;
    private int level;//level in the hierarchy, the toppest is 0
    private int id;
    private int parendId;//the id of direct parent element, the value
                        //will be -1 if the element has no parent element
    private boolean hasChildren;//whether has children
```

```
    private boolean isExpanded; //whether has expanded
    //define two constants, the most top element level is 0, the parent
    //element Id is -1
    public static final int NO_PARENT = -1;
    public static final int TOP_LEVEL = 0;
    public Element(String text, int level, int id, int parendId,
            boolean hasChildren, boolean isExpanded) {
        this.text = text;
        this.level = level;
        this.id = id;
        this.parendId = parendId;
        this.hasChildren = hasChildren;
        this.isExpanded = isExpanded;
    }
    //generates the appropriate set and get methods to set and set
    //corresponding attribute value
    public boolean isExpanded() {
        return isExpanded;
    }
    public void setExpanded(boolean isExpanded) {
        this.isExpanded = isExpanded;
    }
    public String getText() {
        return text;
    }
    public void setText(String text) {
        this.text = text;
    }
    public int getLevel() {
        return level;
    }
    public void setLevel(int level) {
        this.level = level;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getParendId() {
        return parendId;
    }
    public void setParendId(int parendId) {
```

```
            this.parendId = parendId;
        }
        public boolean isHasChildren() {
            return hasChildren;
        }
        public void setHasChildren(boolean hasChildren) {
            this.hasChildren = hasChildren;
        }
}
```

Program code: ProductCatagories\src\iet\jxufe\cn\android\productcatagories\MainActivity.java

```
public class MainActivity extends Activity {
        private ListView mListView;
        private MyAdapter myAdapter;
        private ArrayList<Element> visibleElements;
       //collection of visible elements
        private ArrayList<Element> allElements;//collection of all elements
        private int basePadding = 20;
       //default gap distance between the upper and lower in left padding
        private int baseSize=2;
       //the default size difference between adjacent level font is 2px
        private int baseHeight=8;
       //the default size difference between adjacent level image is 8dp
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            init();
            mListView = (ListView) findViewById(R.id.mListView);
            myAdapter = new MyAdapter();
            mListView.setAdapter(myAdapter);
            mListView.setOnItemClickListener(new MyItemClickListener());
        }
        private void init() {//execute initialization, analog data
            visibleElements = new ArrayList<Element>();
            allElements = new ArrayList<Element>();
            //simulate information of list items
            Element e1 = new Element("Food and Drink", Element.TOP_LEVEL,
                    0,Element.NO_PARENT,true, false);
            Element e2 = new Element("Imported Food", Element.TOP_LEVEL +
                    1, 1, e1.getId(),true, false);//add a first layer node
            Element e3 = new Element("Cookies and cakes", Element.TOP_LEVEL
                    + 2, 2, e2.getId(),true, false);//add a second layer node
            Element e4 = new Element("Sandwich Biscuit", Element.TOP_LEVEL
                    + 3, 3, e3.getId(),false, false);//add a third layer node
```

```
        Element e5 = new Element("Local specialty", Element.TOP_LEVEL + 1, 4, e1.getId(),true, false);//add a first layer node
        Element e6 = new Element("Northwest Region", Element.TOP_LEVEL + 2, 5, e5.getId(),true, false);//add a second layer node
        Element e7 = new Element("Nuts", Element.TOP_LEVEL + 3, 6, e6.getId(),false, false);//add a third layer node
        Element e8 = new Element("Snacks", Element.TOP_LEVEL + 1, 7, e1.getId(),false, false);//add a first layer node.
        Element e9 = new Element("Domestic Appliance", Element.TOP_LEVEL, 8,Element.NO_PARENT,true, false);
        //add an outermost node.
        Element e10 = new Element("Life Appliance", Element.TOP_LEVEL + 1, 9, e9.getId(),true, false);//add a first layer node.
        Element e11 = new Element("Warming Appliance", Element.TOP_LEVEL +2,10,e10.getId(),true, false);//add a second layer node
        Element e12 = new Element("Hand Warmer", Element.TOP_LEVEL + 3, 11,e11.getId(), true, false);//add a third layer node
        Element e13 = new Element("Midea Brand", Element.TOP_LEVEL + 4, 12,e12.getId(), false, false);//add a fourth layer node
        //add the list items to the collection
        allElements.add(e1);
        allElements.add(e2);
        allElements.add(e3);
        allElements.add(e4);
        allElements.add(e5);
        allElements.add(e6);
        allElements.add(e7);
        allElements.add(e8);
        allElements.add(e9);
        allElements.add(e10);
        allElements.add(e11);
        allElements.add(e12);
        allElements.add(e13);
        //add the initial elements to display
        visibleElements.add(e1);
        visibleElements.add(e9);
    }
    private class MyItemClickListener implements OnItemClickListener {
        public void onItemClick(AdapterView<?> parent, View view, int position,
                long id) {
            //get the element which represent the clicked option
            Element element = (Element) myAdapter.getItem(position);
            //check whether the item has children
            if (!element.isHasChildren()) {
```

```
                return;
            }
            if (element.isExpanded()) {
                //remove some elements when switch from spread to close
                element.setExpanded(false);
                //delete corresponding sub-noda data inside this node
                ArrayList<Element> elementsToDel = new
                ArrayList <Element>();
                for (int i = position + 1; i < visibleElements.size();
                    i++) {
                    //exit when meet the same level item.
                    //Otherwise, add the related elements to the
                    //collection to delete.
                    if (element.getLevel() >=
                        visibleElements.get(i).getLevel())
                        break;
                    elementsToDel.add(visibleElements.get(i));
                }
                //delete all the elements to be deleted.
                visibleElements.removeAll(elementsToDel);
                myAdapter.notifyDataSetChanged();
            } else {//add some elements when switch from close to spread.
                element.setExpanded(true);
                //extract sub-elements from the data source and add to
                //the list
                int i = 1;//in order to ensure validity, the counter must
                      //be placed outside the for loop
                //iterate the element in collection
                for (Element e : allElements) {
                    if (e.getParendId() == element.getId()) {
                        e.setExpanded(false);
                        visibleElements.add(position + i, e);
                        i++;
                    }
                }
                myAdapter.notifyDataSetChanged();
            }
        }
    }
    private class MyAdapter extends BaseAdapter {//custom class Adapter.
        //get the count of items in the list
        public int getCount() {
            return visibleElements.size();
        }
```

```
        //get the object in position
        public Object getItem(int position) {
            return visibleElements.get(position);
        }
        //get the Id of item inposition
        public long getItemId(int position) {
            return 0;
        }
        //get the view of item in position
        public View getView(int position, View convertView, ViewGroup
                            parent) {
            //convert the layout file to View object to display
            //information of each item
            View view = getLayoutInflater().inflate(R.layout.item,
            null);
            ImageView icon = (ImageView) view.findViewById(R.id.icon);
            TextView text = (TextView) view.findViewById(R.id.text);
            //get the element of current position.
            Element element = visibleElements.get(position);
            int level = element.getLevel();
            //set the margin of icon
            icon.setPadding(basePadding * level, 0,    0, 0);
            icon.setLayoutParams(new
                    LinearLayout.LayoutParams(LayoutParams.
                    WRAP_CONTENT,40-level*baseHeight));
            text.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    text.getTextSize()-level*baseSize);
            text.setText(element.getText());
            //display icon state
            if (!element.isHasChildren()) {
                //if the element has no child elements, do not display
                //any icon
                Icon.setImageResource(R.drawable.close);
                Icon.setVisibility(View.INVISIBLE);
            } else {//if the element has child elements, display icon
                Icon.setVisibility(View.VISIBLE);
                if (element.isExpanded()) {
                    //check whether the element has spread, if spread,
                    //display expand icon
                    icon.setImageResource(R.drawable.open);
                } else {
                //if the element does not spread, display close icon
                    icon.setImageResource(R.drawable.close);
                }
            }
```

```
130                 return view;
131             }
132         }
133 }
```

15.3 Code Analysis

In this case, we use the multilevel list to realize the ListView effect; the key point is to create an Adapter and the click event handler in each item list. It is still a list substantially, but we dynamically set its left margin, icon size, and text size in the process of building Adapter according to the item's level. And select which item looks like a multilevel list. Then we check if the item contains children item in the process of handling, check if the item contains children. If it contains, then we continue to check whether it is spread or not, and according to our requirement, we remove or add items from the list dynamically.

There is a certain relationship between the list items, which the previous ListView do not have. ID associates the items. Besides the top-level elements, every element in the list has a direct parent element that has the same structure with it. The specific information of each item includes: text, ID, parent element ID, whether has a sub-element, whether to spread. Here, we use an Element class to encapsulate these informations. See codes in Element.java.

The Adapter specified the data displayed in ListView. We need to set the way displayed the data specifically according to the state of list items, which can not realized by the Adapter provided by system, so we need a custom Adapter. In the custom Adapter, the method about displaying list item is getView(). In this method, firstly we need to convert the list item layout file into the corresponding view. Secondly, we can get the corresponding element object according to the location. Thirdly, after getting the object, we can dynamically set its left margin, icon size, and text size according to its hierarchy. Fourthly, we need to check if the element object has a child element. If not, the icon will not be showed, if it has a child element, then show the icon and continue judge which icon will be displayed. If the element is already spread, then display the expanded icon, else display the closed icon. See codes: from 105th-131th line.

During the process of handling list item click event, it is also necessary to check whether the element contains a child element. If not, there is no effect when clicking. If it contains, we need to judge the state of the item when clicking. If it is spread, convert it to close state and hide its sub-elements. If it is closed, convert it to open state and display its sub-element. The state can be switched between spread and closed by clicking, and the data in the list will be updated consistently.

15.4 Extension of Knowledge

The data of this list item in this case is simulated in the program temporarily; it could not be saved permanently. Actually, in most cases, the data is stored in the database, especially for the applications that have CRUD operations. It is different to judge whether the item has a child. We query the database to find a record, which the parentId is the same with the current element's ID. If exists, the current element has a child, otherwise, it does not have a child. Moreover, the operation of judging one item is spread or closed is always needed and the data is real-time changed. There is little value to save the data into the database, and if it is saved in the database, it will affect performance. So all the items are default closed here.

Above all, the following data are all stored in a database, then get the relevant data from the database dynamically and displayed in the list. All the layout files are unchanged and the function is the same. There are some changes in Element.java. Codes showed as follow:

Custom list item: 15\ ProductCatagoriesExt\src\iet\jxufe\cn\android\productcatagoriesext\Element.java

```
public class Element {
    private String text;
    private int level;
    private int id;
    private int parendId;
    private boolean isExpanded;
    //define two constant, the top element is 0,the parentId is -1
    public static final int NO_PARENT = -1;
    public static final int TOP_LEVEL = 0;
    public Element(String text, int level, int id, int parendId) {
        this.text = text;
        this.level = level;
        this.id = id;
        this.parendId = parendId;
    }
    public Element(){}//No argument constructor
    //generate the corresponding set method() and get method()
    public boolean isExpanded() {
        return isExpanded;
    }
    public void setExpanded(boolean isExpanded) {
        this.isExpanded = isExpanded;
    }
    public String getText() {
        return text;
    }
```

```
    public void setText(String text) {
        this.text = text;
    }
    public int getLevel() {
        return level;
    }
    public void setLevel(int level) {
        this.level = level;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getParendId() {
        return parendId;
    }
    public void setParendId(int parendId) {
        this.parendId = parendId;
    }
}
```

Databsehelper: 15\ProductCatagoriesExt\src\iet\jxufe\cn\android\productcatagoriesext\MyOpenHelper.java

```
public class MyOpenHelper extends SQLiteOpenHelper {
    public String createTableSQL = "create table if not exists element_tb"
            + "(_id integer primary key autoincrement,id,text, level, parentId)";
    public MyOpenHelper(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);
    }
    //invoked the method when the database created
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(createTableSQL);
        init(db);
    }
    //call back this method  when updating the database
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        System.out.println("version changed:" + oldVersion + "------>" + newVersion);
```

```
    }
    public void init(SQLiteDatabase db) {
        ArrayList<Element> list = new ArrayList<Element>();
        Element e1 = new Element("Food and Drink ", Element.TOP_LEVEL,
                0,Element.NO_PARENT);
        Element e2 = new Element("Imported Food", Element.TOP_LEVEL + 1,
                1, e1.getId());//add the first level element
        Element e3 = new Element("Cookies and cakes", Element.TOP_LEVEL
                + 2, 2,e2.getId());//add the second level element
        Element e4 = new Element("Sandwich Biscuit", Element.TOP_LEVEL
                +3, 3, e3.getId());//add the third level element
        Element e5 = new Element("Local specialty", Element.TOP_LEVEL +
                1, 4, e1.getId());//add the first level element
        Element e6 = new Element("Northwest Region", Element.TOP_LEVEL
                +2,5,e5.getId());//add the second level element
        Element e7 = new Element("Nuts", Element.TOP_LEVEL + 3, 6,
                e6.getId());    //add the third level element
        Element e8 = new Element("Snacks", Element.TOP_LEVEL + 1, 7,
                e1.getId());//add the first level element
        Element e9 = new Element("Domestic Appliance", Element.TOP_
                LEVEL, 8,Element.NO_PARENT);//
        Element e10 = new Element("Life Appliance", Element.TOP_LEVEL +
                1, 9, e9.getId());//add the first level element
        Element e11 = new Element("Warming Appliance", Element.TOP_LEVEL
                +2,10,e10.getId());//add the second level element
        Element e12 = new Element("Hand Warmer", Element.TOP_LEVEL + 3,
                11,e11.getId());//add the third level element
        Element e13 = new Element("Midea Brand", Element.TOP_LEVEL + 4,
                12,e12.getId());//add the fourth level element
        list.add(e1);
        list.add(e2);
        list.add(e3);
        list.add(e4);
        list.add(e5);
        list.add(e6);
        list.add(e7);
        list.add(e8);
        list.add(e9);
        list.add(e10);
        list.add(e11);
        list.add(e12);
        list.add(e13);
        for (Element element : list) {
            db.execSQL("insert into element_tb(id,text, level,parentId)
                values(?,?,?,?)",
```

```
                    new String[] {element.getId()+"", element.getText(),
                    element.getLevel()+"", element.getParendId()+ ""});
            }
        }
}
```

Program code:
15\ProductCatagoriesExt\src\iet\jxufe\cn\android\productcatagoriesext\MainActivity.java

```
public class MainActivity extends Activity {
    private ListView mListView;
    private MyAdapter myAdapter;
    private ArrayList<Element> visibleElements;
    private int basePadding = 20;//default gap distance between the upper
                            //and lower in left padding
    private int baseSize=2;//the default size difference between
                        //adjacent level font is 2px
    private int baseHeight=8;//the default size difference between
                        //adjacent level image is 8dp
    private MyOpenHelper mHelper;
    private SQLiteDatabase mDB;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mListView = (ListView) findViewById(R.id.mListView);
        mHelper=new MyOpenHelper(this, "element.db", null,1);
        mDB=mHelper.getWritableDatabase();//get the database
        visibleElements=getData("select * from element_tb where parented
                =?", new String[]{Element.NO_PARENT+""});
        myAdapter = new MyAdapter();//create a Adapter
        mListView.setAdapter(myAdapter);
        mListView.setOnItemClickListener(new MyItemClickListener());
        //add a ClickListener for the ListView item
    }
    private ArrayList<Element> getData(String sql,String[] args){
        //get the query result by the condition
        ArrayList<Element> list=new ArrayList<Element>();
        Cursor cursor=mDB.rawQuery(sql, args);
        while(cursor.moveToNext()){
            Element element=new Element();//create element
            element.setId(cursor.getInt(cursor.getColumnIndex("id")));
            element.setLevel(cursor.getInt(cursor.getColumnIndex
                    ("level")));
            element.setParendId(cursor.getInt(cursor.getColumnIndex
                    ("parentId")));
```

```
                    element.setText(cursor.getString(cursor.getColumnIndex
                                              ("text")));
                    element.setExpanded(false);
                         //by default, each element is not expanded
                    list.add(element);//put all the data into the list
               }
               return list;
          }
          private boolean hasChildren(Element element){
                //check whether the element has a child element
               ArrayList<Element> list=getData("select * from element_tb where
parentId=?", new String[]{element.getId()+""});
               if(list!=null&&list.size()!=0){
                    return true;
               }else return false;
          }
          private class MyItemClickListener implements OnItemClickListener {
               public void onItemClick(AdapterView<?> parent, View view, int
position,
                         long id) {
                    //get the item which clicked
                    Element element = (Element) myAdapter.getItem(position);
                    //judge wheather the item has child or not, if not,return.
                    //or else judge whether the item is expanded or not
                    if (!hasChildren(element)) {
                         return;
                    }
                    if (element.isExpanded()) {
                         //from expanded to closed should delete some elements.
                         element.setExpanded(false);
                         //delete the element and its children.
                         ArrayList<Element>elementsToDel=new ArrayList <Element>();
                         for (int i = position + 1; i < visibleElements.size(); i++){
                              //if the level of the element equals the level of
                              //current element
          //should exit the loop. Else should add the element into the list
          //which need to delete
                              if (element.getLevel() >=
                                                   visibleElements.get(i).getLevel())
                                   break;
                              elementsToDel.add(visibleElements.get(i));
                         }
                         //delete all the elements which need delete
                         visibleElements.removeAll(elementsToDel);
                         myAdapter.notifyDataSetChanged();
```

```
                } else {//from closed to expanded should add some elements
                    element.setExpanded(true);
                    //query the child element from the database
                    ArrayList<Element> addElements=getData("select * from
                            element_tb where parentId=?", new String[]
                            {element.getId()+""});
                    visibleElements.addAll(position+1,addElements);
                    myAdapter.notifyDataSetChanged();
                }
            }
        }
    private class MyAdapter extends BaseAdapter {//create a custom adapter
        public int getCount() {//get the count of items in the list
            return visibleElements.size();
        }
        public Object getItem(int position) {//get the object in position
            return visibleElements.get(position);
        }
        public long getItemId(int position) {
         //get the Id of item inposition
            return 0;
        }
        //get the view of item in position
        public View getView(int position, View convertView, ViewGroup
parent) {
            //convert the layout file to View object to display
information of each item
            View view = getLayoutInflater().inflate(R.layout.item, null);
            ImageView icon = (ImageView) view.findViewById(R.id.icon);
            TextView text = (TextView) view.findViewById(R.id.text);
            //get the element of current position
            Element element = visibleElements.get(position);
            int level = element.getLevel();
            //set the margin of icon
            icon.setPadding(basePadding * level, 0,    0, 0);
            icon.setLayoutParams(new LinearLayout.LayoutParams
                    (LayoutParams.WRAP_CONTENT,40-level*baseHeight));
            text.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                        text.getTextSize()-level*baseSize);
            text.setText(element.getText());
            //display icon state
            if (!hasChildren(element)) {
                //if the element has no child elements, do not display
                icon.setImageResource(R.drawable.close);
```

```
                icon.setVisibility(View.INVISIBLE);
            } else {//if the element has child elements, display icon
                icon.setVisibility(View.VISIBLE);
                if (element.isExpanded()) {
                    //check whether the element has spread, if spread,
                      display expand icon
                    icon.setImageResource(R.drawable.open);
                } else {//if the element does not spread, display close icon
                    icon.setImageResource(R.drawable.close);
                }
            }
            return view;
        }
    }
    protected void onDestroy() {//close the database when destroy
        if(mDB!=null){
            mDB.close();
        }
        super.onDestroy();
    }
}
```

15.5 Thinking and Exercises

Try to add a long press event handler for the list item. Such popups a dialog after long press and allows the user to select the operations, delete the item, and add a sub-element for it. The item and its child items will be deleted when the user select to delete the item. The user can set the name and other attributes of sub-element when he selects to add a sub-element for it.

Chapter 16 College Introduction—TabHost

16.1 Case Overview

This case mainly implements the function of displaying college information. We show the college information from multiple aspects through different tabs and it is very easy to switch to other page from one page. This case uses a combination of TabHost and Fragment, the entire application contains one main Activity only. It should also change the page information dynamically when switching pages. TabHost is commonly used in applications, which has a navigation function in it. In this case, we introduced the school information from followed three aspects: school introduction, current leadership, and department information. Our application looks like Figure 16-1 when running.



Figure 16-1 the figure of the running results at different fragments

16.2 Key Code

Main interface layout file: 16\ CollegeInfo\res\layout\activity_main.xml

```
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:id="@+id/mTabHost"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc" >
    <!—LinearLayout overall，include two parts of tabs and specific page display -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
        <!-- display information of a single page-->
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1" >
            <FrameLayout
                android:id="@+id/realcontent"
                android:layout_width="match_parent"
                android:layout_height="match_parent" />
        </FrameLayout>
        <!—display all the tabs at the bottom -->
        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#66666666" >
        </TabWidget>
    </LinearLayout>
</TabHost>
```

Each tab contains two pieces of information : icon and title, layout information as follow:

Single tab layoyut file: 16\CollegeInfo\res\layout\tab.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:background="@drawable/bg"
    android:padding="5dp"
    android:orientation="vertical">
    <ImageView
```

```
        android:id="@+id/icon"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:contentDescription="@string/imgInfo" />
    <TextView
        android:id="@+id/title"
        android:textColor="#0000ff"
        android:textSize="12sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Background picture of tab: 16\ CollegeInfo\res\drawable-hdpi\bg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <!—The picture uder selected state is different from the picture uder other states. -->
    <item android:state_selected="true" android:drawable = "@drawable/bg_choosed"/>
    <item android:state_pressed="true" android:drawable = "@drawable/bg_choosed"/>
    <item  android:drawable="@drawable/bg_unchoosed"/>
</selector>
```

The corresponding code of two custom background image:

Picture when selected: 16\ CollegeInfo\res\drawable-hdpi\ bg_choosed.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" >
    <solid android:color="#554455"/>
    <stroke android:width="1dp"
        android:color="#aa0000"/>
</shape>
```

Picture when unselected: 16\ CollegeInfo\res\drawable-hdpi\ bg_unchoosed.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" >
    <solid android:color="#666666"/>
    <stroke android:width="1dp"
        android:color="#000055"/>
</shape>
```

Layout file of college introduction and its main routine code:

Layout file of college introduction page: 16\ CollegeInfo\res\layout\college.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc"
    android:orientation="vertical">
   <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:gravity="center"
       android:textSize="24sp"
       android:background="#77ccbbaa"
       android:padding="10dp"
       android:text="@string/collegeTitle"/>
   <ScrollView
       android:layout_width="match_parent"
       android:layout_height="wrap_content" >
   <TextView
       android:id="@+id/infoView"
       android:textSize="16sp"
       android:textColor="#004400"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:padding="5dp"/>
   </ScrollView>
</LinearLayout>
```

Page routine: 16\CollegeInfo\src\iet\jxufe\cn\android\collegeinfo\CollegeInfoFragment.java

```
public class CollegeInfoFragment extends Fragment {
    private TextView infoView;
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        //convert layout file into a View object
        View collegeView = inflater.inflate(R.layout.college,
            container, false);
        infoView = (TextView) collegeView.findViewById(R.id.infoView);
        //save the college information in a text file and read it through
        //IO stream
        InputStream inputStream=getResources().openRawResource
          (R.raw.college_info);
        infoView.setText(getStringFromInputStream(inputStream));
        return collegeView;
    }
```

```
    //read string from the input stream
    public String getStringFromInputStream(InputStream inputStream) {
        byte[] buffer = new byte[1024];
        int hasRead = 0;//record the number of bytes have read
        StringBuilder result = new StringBuilder("");
        try {
            while ((hasRead = inputStream.read(buffer)) != -1) {
                //build a string based on the read bytes and add it to
                //the end of the existing string
                result.append(new String(buffer, 0, hasRead, "GBK"));
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return result.toString();
    }
}
```

Layout file of current leader page and its main routine code:

Page layout file: 16\ CollegeInfo\res\layout\leader.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc"
    android:orientation="vertical">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="24sp"
        android:background="#77ccbbaa"
        android:padding="10dp"
        android:text="@string/leaderTitle"/>
    <ListView
        android:id="@+id/mListView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:divider="#666666"
        android:dividerHeight="2dp"/>
</LinearLayout>
```

Layout file of single item in the list: 16\ CollegeInfo\res\layout\item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/image"
        android:layout_width="60dp"
        android:layout_height="72dp"
        android:layout_margin="5dp"
        android:contentDescription="@string/imgInfo" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:gravity="center_vertical"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#000000"
            android:textSize="20sp" />
        <TextView
            android:id="@+id/job"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:gravity="left"
            android:textColor="#0000ee"
            android:textSize="16sp" />
    </LinearLayout>
</LinearLayout>
```

Page routine: 16\CollegeInfo\src\iet\jxufe\cn\android\collegeinfo\LeaderFragment.java

```
public class LeaderFragment extends Fragment {
    private ListView mListView;
    private String[] names = new String[] { "Aihao Guan", "Xinhai Li",
            "Maojun Huang", "Yaohui Bai", "Qingshan Deng","Min Peng" };
    private int[] imgIds = new int[] { R.drawable.guanaihao,
            R.drawable.lixinhai, R.drawable.huangmaojun,
            R.drawable.baiyaohui,
            R.drawable.dengqingshan, R.drawable.pengmin };
    private String[] jobs = new String[] { "dean", "secretary", "vice dean
            on teaching", "vice dean on scientific research","vice dean
            on subject construction", "vice secretary"};
```

```
    private List<Map<String, Object>> list = new ArrayList<Map<String, Object>>();
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View leaderView=inflater.inflate(R.layout.leader, container, false);
        mListView=(ListView)leaderView.findViewById(R.id.mListView);
        Init();
        SimpleAdapter adapter = new SimpleAdapter(getActivity(), list, R.layout.item,
                new String[] { "name", "img", "job" }, new int[] { R.id.name,
                        R.id.image, R.id.job });
        mListView.setAdapter(adapter);
        return leaderView;
    }
    public void init() {
        for (int i = 0; i < names.length; i++) {
            Map<String, Object> item = new HashMap<String, Object>();
            item.put("name", "name: "+names[i]);
            item.put("img", imgIds[i]);
            item.put("job", "job: "+jobs[i]);
            list.add(item);
        }
    }
}
```

Layout file of department page and its main routine code:

Page layout file: 16\CollegeInfo\res\layout\department.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc"
    android:orientation="vertical">
   <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:gravity="center"
       android:textSize="20sp"
       android:background="#77ccbbaa"
       android:padding="10dp"
       android:text="@string/departmentTitle"/>
   <ListView
       android:id="@+id/departmentView"
```

```
        android:textSize="12sp"
        android:textColor="#aa0000aa"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"/>
</LinearLayout>
```

Page routine: 16\CollegeInfo\src\iet\jxufe\cn\android\collegeinfo\DepartmentFragment.java

```
public class DepartmentFragment extends Fragment {
    private ListView mListView;
    private String[] names = new String[] { "Electronic Engineering",
            "Software Engineering", "Network Engineering",
            "Communication Engineering" };
    public View onCreateView(LayoutInflater inflater, ViewGroup
            container,
            Bundle savedInstanceState) {
        View leaderView=inflater.inflate(R.layout.department,
             container, false);
        mListView=(ListView)leaderView.findViewById
             (R.id.departmentView);
        ArrayAdapter<String> adapter=new ArrayAdapter <String>
           (getActivity(), android.R.layout.simple_list_item_1,names);
        mListView.setAdapter(adapter);
        return leaderView;
    }
}
```

Main routine: 16\CollegeInfo\src\iet\jxufe\cn\android\collegeinfo\MainActivity.java

```
public class MainActivity extends Activity {
    private TabHost mTabHost;
    private int[] icons=new int[]{R.drawable.college, R.drawable. leader,
                        R.drawable.department};
    private String[] tags=new String[]{"college","leader", "department"};
    private String[] titles=new String[]{"college","leader", "department" };
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);//hide the title
bar.
        setContentView(R.layout.activity_main);
        mTabHost = (TabHost) findViewById(R.id.mTabHost);
        mTabHost.setup();
        for(int i=0;i<titles.length;i++){//add tabs in loop
            TabSpec tabSpec=mTabHost.newTabSpec(tags[i]);
            //create an option, and specify its tag
```

```
                View view=getLayoutInflater().inflate(R.layout.tab, null);
                TextView titleView=(TextView)view.findViewById(R.id.title);
                ImageView iconView=(ImageView)view.findViewById(R.id.icon);
                titleView.setText(titles[i]);
                iconView.setImageResource(icons[i]);
                tabSpec.setIndicator(view);
                tabSpec.setContent(R.id.realcontent);
                mTabHost.addTab(tabSpec);
            }
            mTabHost.setOnTabChangedListener(new MyTabChangedListener());
            mTabHost.setCurrentTabByTag("leader");//set the tab page when
                                                   //initialization
        }
        private class MyTabChangedListener implements OnTabChangeListener {
            //custom tab change event listener
            public void onTabChanged(String tabTag) {
                FragmentTransaction fragmentTransaction=
                getFragmentManager().beginTransaction();
                //judge which tab page is clicked
                if(tabTag.equalsIgnoreCase("leader")){
                    fragmentTransaction.replace(R.id.realcontent,
                        new LeaderFragment(), "leader");
                }else if(tabTag.equalsIgnoreCase("college")){
                    fragmentTransaction.replace(R.id.realcontent,
                        new CollegeInfoFragment(), "college");
                }else if(tabTag.equalsIgnoreCase("department")){
                    fragmentTransaction.replace(R.id.realcontent,
                        new DepartmentFragment(), "department");
                }
                fragmentTransaction.commit();
            }
        }
    }
```

16.3 Code Analysis

16.3.1 TabHost introduction

TabHost is a common used widget in Android applications, commonly used in page navigation and page switch. user can easily switch between multiple pages in an Activity with it. It mainly consists two parts: TabWidget and TabContent. TabWidget mainly used to display different options such as school introduction, current leadership, and department information in

this case. It can be placed at the top of page or at the bottom of the page according to the user's requirement.

TabSpec represents each option in TabHost, you can set tags, icons, titles, and correspond content for it. After clicking a TabSpec, the corresponding information will be displayed in the TabContent. TabSpec is an internal class of TabHost, it doesn't provide an outward public constructor, and therefore it cannot be instantiated by using the keyword new. It needs to call the method newSpec() of TabHost to create a TabSpec. Then do some simple setup, finally add it to the TabHost, specific codes: from 13th-21th line.

Similar to the ListView, there are two ways to use TabHost: the first way is placing a TabHost widget in the layout file, then find it by ID and do the related operation; The second way is inheriting from a class supplied by system – TabActivity. At this point the page will automatically include a TabHost. We can get this TabHost through the method getTabhost() of TabActivity, then do some related operation. The second way is more convenient. However, TabActivity has been abandoned in Android API Level 13, so Android official document recommend us to use Fragment. This case describes how to achieve switching function through the custom TabHost in the layout.

After defining the TabHost in the layout file, you need to add two widgets in it : TabWidget widget and FrameLayout widget, add IDs for them. And the value is certain there are the system-defined constant **android:id/tabs** and **@android:id/tabcontent** respectively. TabHost has no requirement about ID. This is because method setUp() will be called before loading TabSpec, and there follow statement as follows inside it:

```
mTabWidget = (TabWidget) findViewById(com.android.internal.R.id.tabs);
if (mTabWidget == null) {
    throw new RuntimeException("Your TabHost must have a TabWidget whose
                            id attribute is 'android.R.id.tabs'");
}
…
mTabContent = (FrameLayout)
findViewById(com.android.internal.R.id.tabcontent);
if (mTabContent == null) {
    throw new RuntimeException("Your TabHost must have a FrameLayout whose
                            id attribute is 'android.R.id.tabcontent'");
}
```

In this method, the system will find TabWidget according to com.android.internal.R.id.tabs, if it is not exist, the widget throws an exception to the user: Your TabHost must have a TabWidget whose ID is "android.R.id.tabs". Find FrameLayout according to com.android.internal.R.id.tabcontent, if it is not exist, it also throws an exception to the user: Your TabHost must have a FrameLayout whose ID is "android.R.id.tabcontent". The position of TabWidget and FrameLayout can be set according to the requirement. In this case, we put the TabWidget at

the bottom of this page and FrameLayout fill all the remaining space. See codes in activity_main.xml.

After selecting an option in TabHost, the page content will dynamically be changed. So we need to add a page change event listener for the TabHost. Once the page changed, the content in the FrameLayout also be changed, this part of content in each page is realized by Fragment.

16.3.2 Fragment introduction

Fragment is a new API, it is first come out in Android 3.0. A Fragment represents a behavior or a portion of user interface in an Activity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities. You can think of a fragment as a modular section of an activity, which has its own life cycle, receives its own input events, and which you can add or remove while the activity is running.

A fragment must always be embedded in an activity and the fragment's life cycle is directly affected by the host activity’s life cycle. For example, when the activity is paused, so do all fragments in it, and when the activity is destroyed, so do all fragments. However, while an activity is running, you can manipulate each fragment independently, such as add or remove them. Main features about Fragment :

- One Activity can have many Fragment, vice versa, a Fragment also can be used by many Activity.
- Fragment is always an intergal part of Activity interface. The fragment can access the Activity instance with getActivity() and easily perform tasks such as find a view in the activity layout. To manage the fragments in your activity, you need to use FragmentManager. To get it, call getFragmentManager() from your activity. Get fragments that exist in the activity, with findFragmentById() or findFragmentByTag().
- Though Fragment defines its own lifecycle, that lifecycle is dependent on its activity: if the activity is stopped, no fragments inside of it can be started; when the activity is destroyed, all fragments will be destroyed.
- Only when the activity is running, you can use method such as add(), remove(), replace() to manipulate Fragment.

To create a fragment, you must create a subclass of Fragment (or an existing subclass of it). The Fragment class has code that looks a lot like an Activity. It contains callback methods similar to an activity, such as onCreate(), onStart(), onPause(), and onStop(). To provide a layout for a fragment, you must implement the onCreateView() callback method, which the Android system calls when it's time for the fragment to draw its layout. Your implementation of this method must return a View that is the root of your fragment's layout. In this case, all the three Fragments are not complicated that we only override method onCreateView().

After Fragment is created, you need to insert the fragment into your activity. There are two

ways for you to do this:

- Declare the fragment inside the activity's layout file by tag <fragment>. Specify layout properties for the fragment as if it were a view. The android:name attribute in the <fragment> specifies the Fragment class to instantiate in the layout.
- Programmatically add the fragment to an existing ViewGroup. To make fragment transactions in your activity (such as add, remove, or replace a fragment), you must use APIs from FragmentTransaction. You can get an instance of FragmentTransaction from your Activity. You can then add a fragment using the add() method, specifying the fragment to add and the view in which to insert it. Once you've made your changes with FragmentTransaction, you must call commit() for the changes to take effect.

In this case, we need to change Fragment dynamically, so we choose the second way. See codes: from 38th-51th line.

16.3.3 Change the picture according to state

In Android applications,it is used to change the background or image of widget according to the state in order to distinguish the user's operation. There are two ways to carry out it : the first one is adding the appropriate event handler for the widget. Then change its background or image in the corresponding method through codes. The second way is defining a special XML picture, you can provide a different background image for each state. The first way is relatively trouble and hard to reuse while the second way just needs us define a XML picture, when there is a widget needs it, reference to it directly. We choose the second way in this case.

We can describe the state list in an XML file. Each graphic is represented by an <item> element inside a single <selector> element. Each <item> uses various attributes to describe the state in which it should be used as the graphic for the drawable. Two main attributes in Item:

- **android:state_xxx**: Assign a special state.
- **android:drawable**: Reference to a drawable resource. Main state in Item element.
- **android:state_active**: Set when a view or its parent has been "activated" meaning the user has currently marked it as being of interest.
- **android:state_checkable**: State identifier indicating that the object may display a check mark.
- **android:state_checked**: State identifier indicating that the object is currently checked.
- **android:state_enabled**: Set when a view is enabled.
- **android:state_first**: Set when a view at the begin state.
- **android:state_focused**: Set when a view has input focus.
- **android:state_last**: Set when a view at the end state.
- **android:state_middle**: Set when a view at the middle state.
- **android:state_pressed**: Set when the user is pressing down in a view.
- **android:state_selected**: Set when a view (or one of its parents) is currently selected.

- **android:state_window_focused**: Set when a view's window has input focus.

The tag <selector> also can be replaced by a Java class and this class is named StateListDrawable. Class StateListDrawable provides us a method addState(int[] stateSet, Drawable drawable) to add a new image/string ID to the set of images which is very similar to tag <item>.

16.4 Extension of Knowledge

16.4.1 Communicating with the activity

The above case is just a brief introduction of Fragment, it does not relate to a Fragment event handler. Next, we will add event handler for the department page which listing item based on the above case. When the item has been chose, the system should show its information in detail. Press back button, return to the department list page. Our application running looks like Figure 16-2.

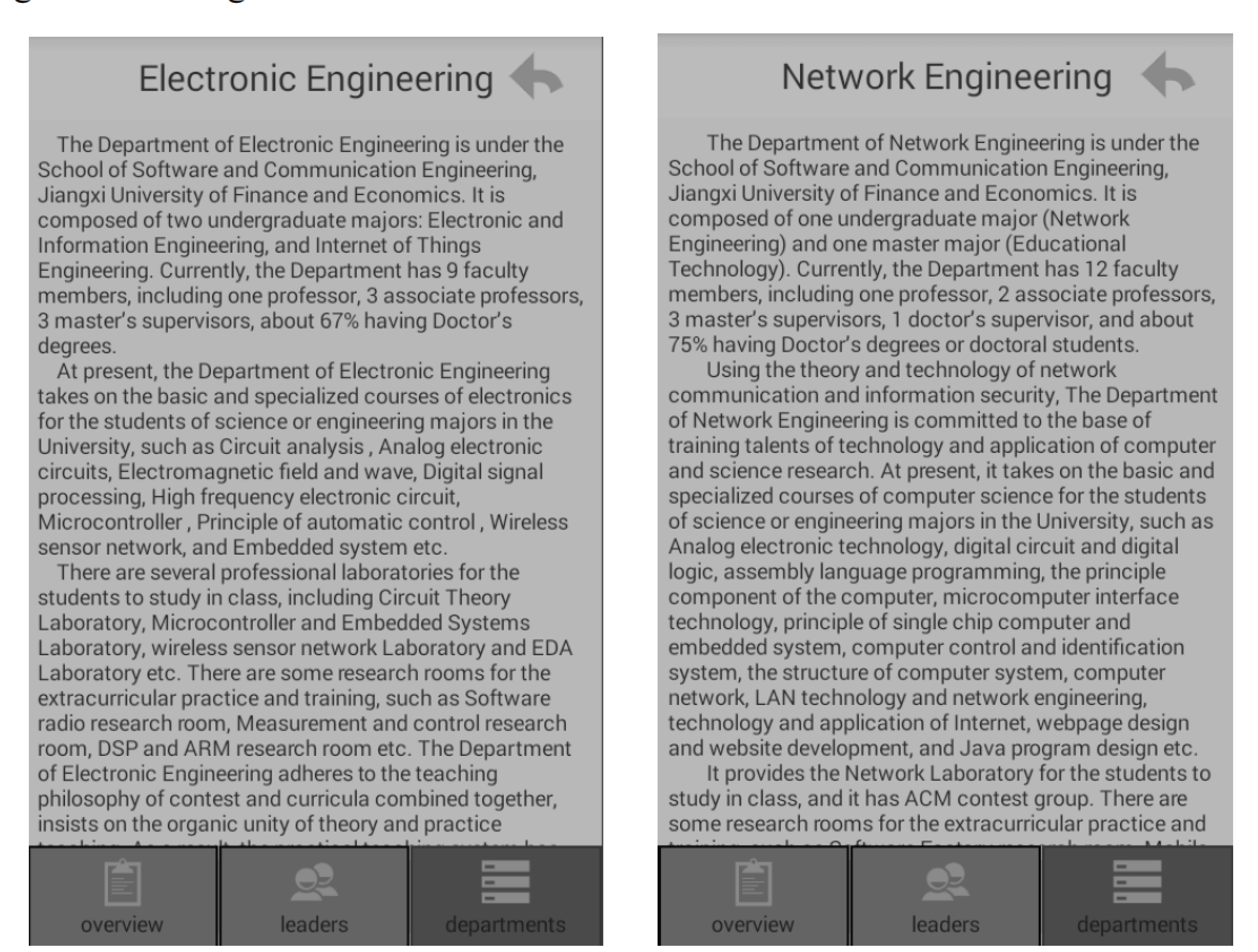
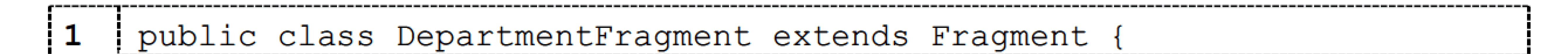


Figure 16-2 the figure of showing detail information of department

Making some changes in DepartmentFragment.java, adding an event handler for each list item. As the introductory text of each department is too much to write in the code directly, so we store it in txt file and save this file under folder res / raw. DepartmentFragment codes:



```
public class DepartmentFragment extends Fragment {
```

```
    private int[] infos = new int[] { R.raw.dianzi, R.raw.wangluo,
            R.raw.ruanjian, R.raw.tongxin, R.raw.peixun };
            //store the detail information of department
    private ListView mListView;
    private String[] names = new String[] { "Electronic Engineering",
            "Software Engineering", "Network Engineering",
            "Communication Engineering" };
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View leaderView = inflater.inflate(R.layout.department,
container, false);
        mListView = (ListView) leaderView.findViewById
          (R.id.departmentView);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>
         (getActivity(),
                android.R.layout.simple_list_item_1, names);
        mListView.setAdapter(adapter);
        mListView.setOnItemClickListener(new MyItemClickListener());
        return leaderView;
    }
    private class MyItemClickListener implements OnItemClickListener {
        public void onItemClick(AdapterView<?> parent, View view, int
position,
                long id) {
            Bundle bundle = new Bundle();//create bundele object to
transfer data
            bundle.putInt("detailId", infos[position]);//transfer title
            bundle.putString("title", names[position]);//transfer the
id of file
            DepartmentInfoFragment depInfoFragment = new
DepartmentInfoFragment();            //create Fragment object
            depInfoFragment.setArguments(bundle);//set parameters.
            getActivity().getFragmentManager().beginTransaction()
                .replace(R.id.realcontent, depInfoFragment).commit();
            //replace the current Fragment with a new Fragment.
        }
    }
}
```

Layout file of detailed department information: 16\CollegeInfo\res\layout\department_info.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#aabbcc"
```

```
    android:orientation="vertical" >
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#77ccbbaa">
        <TextView
            android:id="@+id/infoTitle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:padding="10dp"
            android:textSize="24sp" />
        <ImageButton
            android:id="@+id/goBack"
            android:src="@drawable/goback"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="#00000000"
            android:layout_marginRight="20dp"
            android:contentDescription="@string/imgInfo"/>
    </RelativeLayout>
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:id="@+id/detailInfo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:textColor="#004400"
            android:textSize="16sp" />
    </ScrollView>
</LinearLayout>
```

Detailed information page:
16\CollegeInfo\src\iet\jxufe\cn\android\collegeinfo\DepartmentInfoFragment.java

```
public class DepartmentInfoFragment extends Fragment {
    private TextView infoTitle, detailInfo;
    private ImageButton goBack;
    private String titleString="Electronic Engineering";//store the
                                                    //title information
    private int detailId=R.raw.dianzi;//store the id of the file
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Bundle bundle = getArguments();
        if(bundle!=null){
            titleString = bundle.getString("title"," Electronic Engineering");
            //get title from the parameter, default is Electronic Engineering
            detailId = bundle.getInt("detailId", R.raw.dianzi);
            //get the id of the file from the parameter
        }
    }
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View detailView = inflater.inflate(R.layout.department_info,
container, false);
        detailInfo=(TextView)detailView.findViewById(R.id.detailInfo);
        infoTitle = (TextView) detailView.findViewById(R.id.infoTitle);
        goBack = (ImageButton) detailView.findViewById(R.id.goBack);
        infoTitle.setText(titleString);
        InputStream inputStream = getResources().openRawResource
(detailId);
        detailInfo.setText(getStringFromInputStream(inputStream));
        goBack.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                getActivity().getFragmentManager().beginTransaction()
                        .replace(R.id.realcontent, new
DepartmentFragment())
                        .commit();
            }
        });
        return detailView;
    }
    //read string from the input stream.
    public String getStringFromInputStream(InputStream inputStream) {
        byte[] buffer = new byte[1080];
        int hasRead = 0;//record the num of the string has read
        StringBuilder result = new StringBuilder("");
        try {
            while ((hasRead = inputStream.read(buffer)) != -1) {
//Construct a string with the read bytes and append to the end of the current
//string
                result.append(new String(buffer, 0, hasRead, "GBK"));
            }
        } catch (Exception ex) {
            ex.printStackTrace();
```

```
            }
            return result.toString();
        }
}
```

When you need to pass parameters in the Fragment, you can create a Bundle packet, then call method setArgument(Bundle bundle) to pass the Bundle packet to the Fragment. Get this packet by using method getArgument() in Fragment, finally do the related operation.

16.4.2 Switch page through ActionBar

In addition to Tabhost, Tabs in the action bar make it easy for users to explore and switch between different views in your app. And it is relatively easy. When our application running, it looked like Figure 16-3.



Figure 16-3 the figure of the running results at different fragments

The content of each page is the same with previous case, still using Fragment. The most difference compared with the last case is the main interface and main program. Specific codes:

Main interface layout file: 16\ ActionBarTab\res\layout\activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:id="@+id/container"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#aabbcc">
</RelativeLayout>
```

Program code: 16\ActionBarTab\src\iet\jxufe\cn\android\actionbartab\MainActivity.java

```
public class MainActivity extends Activity {
    private ActionBar mActionBar;
    private String[] titles=new String[]{"college","leader", "department"};
    //titles of the tabs
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mActionBar=getActionBar();//get the ActionBar
        mActionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
        //set the mode of ActionBar as navigation.
        MyTabListener mTabListener=new MyTabListener();
        //create a TabListener object
        for(int i=0;i<titles.length;i++){//add tabs to ActionBar in a loop
            Tab tab=mActionBar.newTab();//create a tab
            tab.setText(titles[i]);//set the title of the tab
            tab.setTabListener(mTabListener);//add a tabListener for
                                              //the tab
            mActionBar.addTab(tab);//add the tab into ActionBar
        }
    }
    private class MyTabListener implements TabListener{
        public void onTabSelected(Tab tab, FragmentTransaction ft) {
            //Event handler when the tab is selected.
            String tabText=tab.getText().toString();
            //get the title of the selected tab
            FragmentTransaction fragmentTransaction =
                        getFragmentManager().beginTransaction();
            if(tabText.equalsIgnoreCase("leader")){
                //replace the selected page with a new Fragment.
                fragmentTransaction.replace(R.id.container, new LeaderFragment());
            }else if(tabText.equalsIgnoreCase("college")){
                fragmentTransaction.replace(R.id.container,new CollegeInfoFragment());
            }else if(tabText.equalsIgnoreCase("department")){
                fragmentTransaction.replace(R.id.container,new DepartmentFragment());
            }
            fragmentTransaction.commit();
        }
        public void onTabUnselected(Tab tab, FragmentTransaction ft) {}
```

```
            //event handler when the tab is not selected.
            public void onTabReselected(Tab tab, FragmentTransaction ft) {}
            //event handler when the tab is selected again.
        }
}
```

We can see through the effect of this case that although Actionbar makes it easy for users to explore and switch between different views, there are also some drawbacks in sides. The ActionBar is fixed by system; it can only be placed at the top of the screen, which is not as flexible as TabHost. By default, the application can not remove the icon and title, the space is very limited and the content of each tab could not be customed at will. Although Tab class has provided us method setIcon() to assign an icon for the tab, but the title and icon could only be placed horizontally while TabSpec can pass a View as tab, this is much more flexible. In summary, they both have advantages and disadvantages, you can choose to use them according to the specific requirement. When the requirement is very easy, we recommended using ActionBar. And when the requirement is complex and requires high flexibility and scalability, we recommended using TabHost.

16.5 Thinking and Exercises

(1) Change the part of switching page in TabHost page, making the options displayed at the top.

(2) Complete the part of switching page in ActionBar page, displaying detailed department information after select an item in the department list.

Chapter 17 The Sound of Music—Music Player

17.1 Case Overview

This case mainly implements the function of music player, playing music is a time-consuming operation, so we usually perform it on the background while the main thread can do other things such as browsing the web. It means that the system can continue playing music even if the music player application has quit. The communication of foreground and background is mainly accomplished by broadcast. When the foreground needs to do some change to the music which is playing in the background like : the first song, previous song, play/pause, next song, the last song. The foreground will send a broadcast, the background will deal it accordingly when it receive the broadcast. When the playing is finished, do the post processing according to the user's setting about play type such as list cycle, single cycle, random play and so on. Besides, this application also allow the user set a piece of music as ring tones. When our application running, it looks like Figure 17-1.

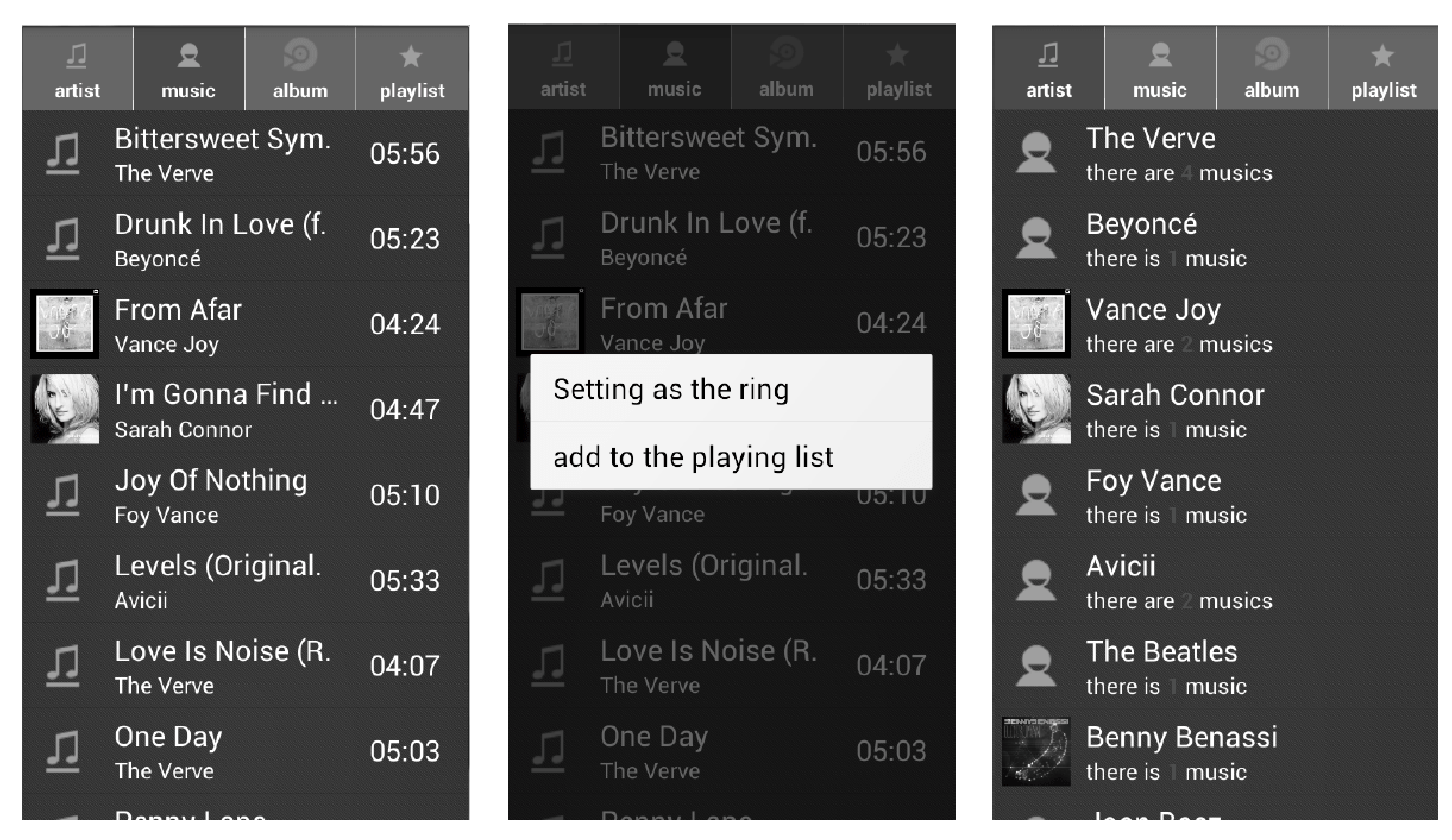
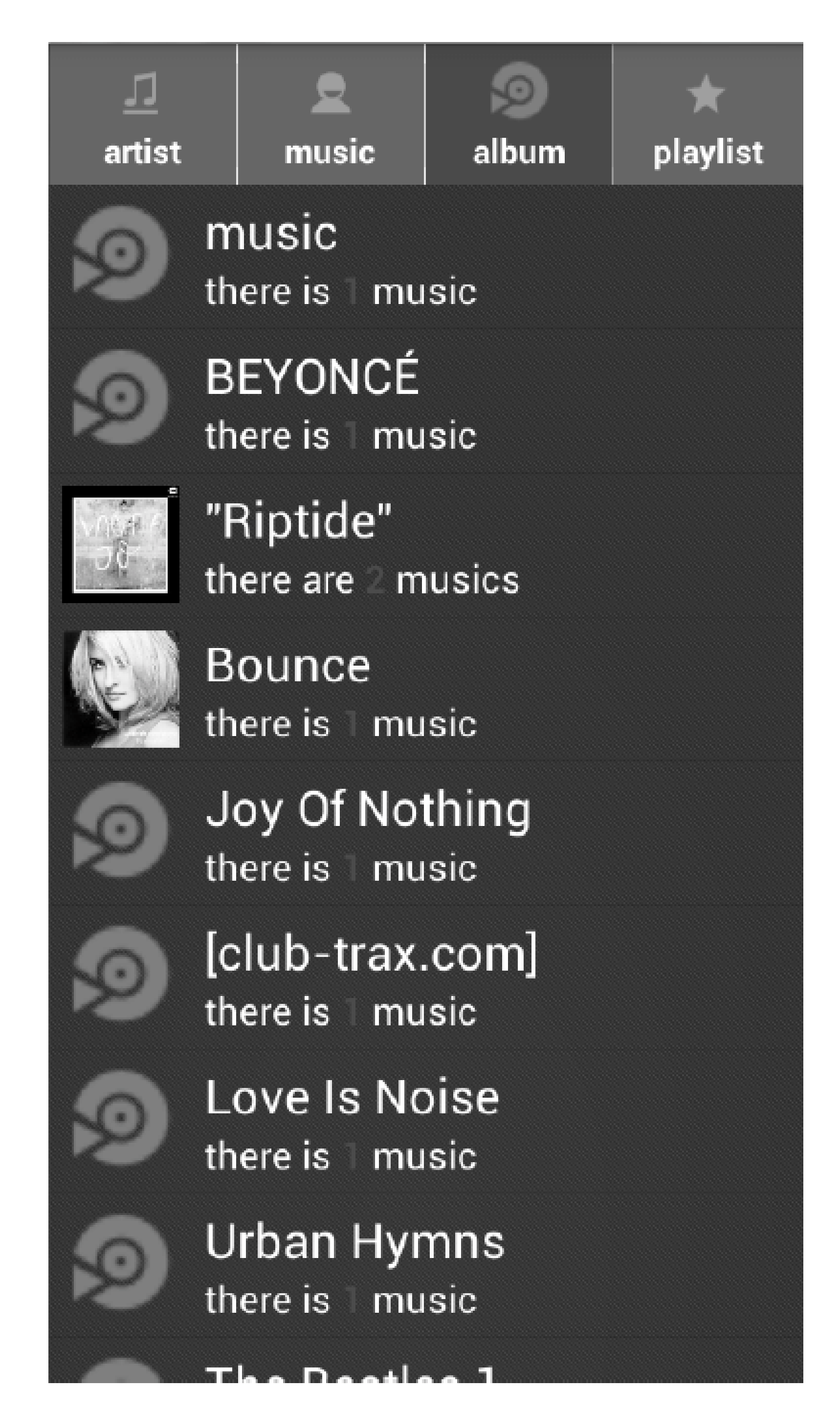
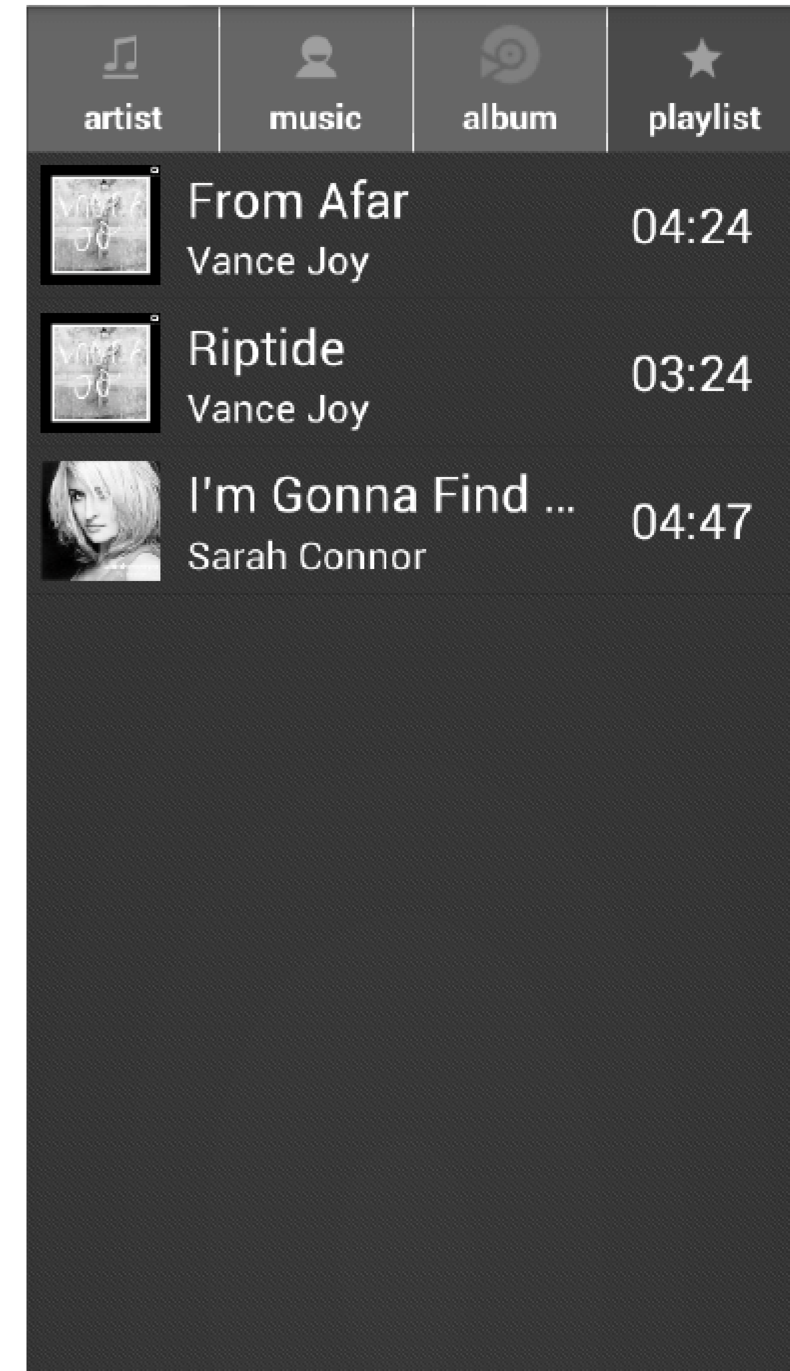
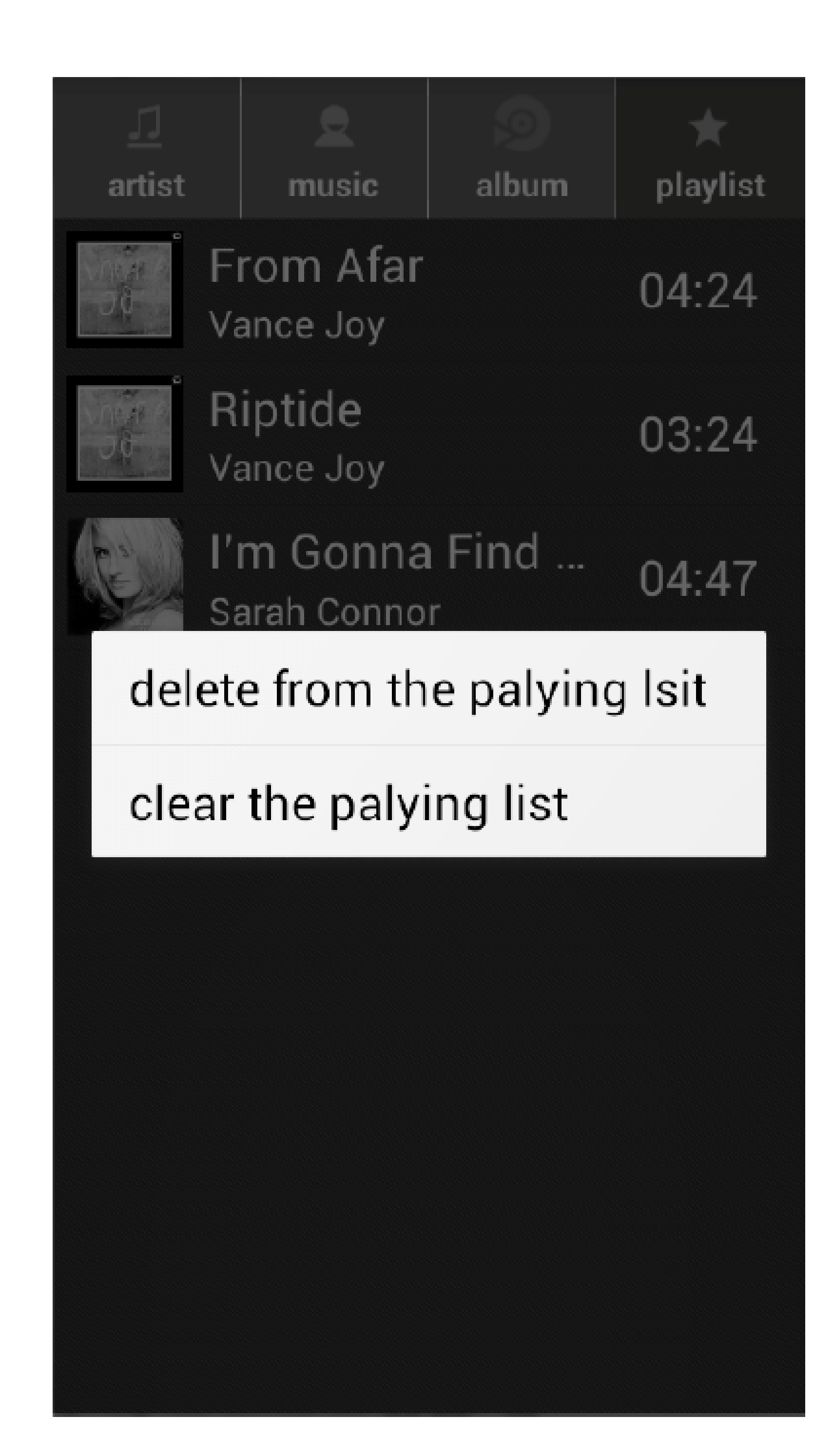
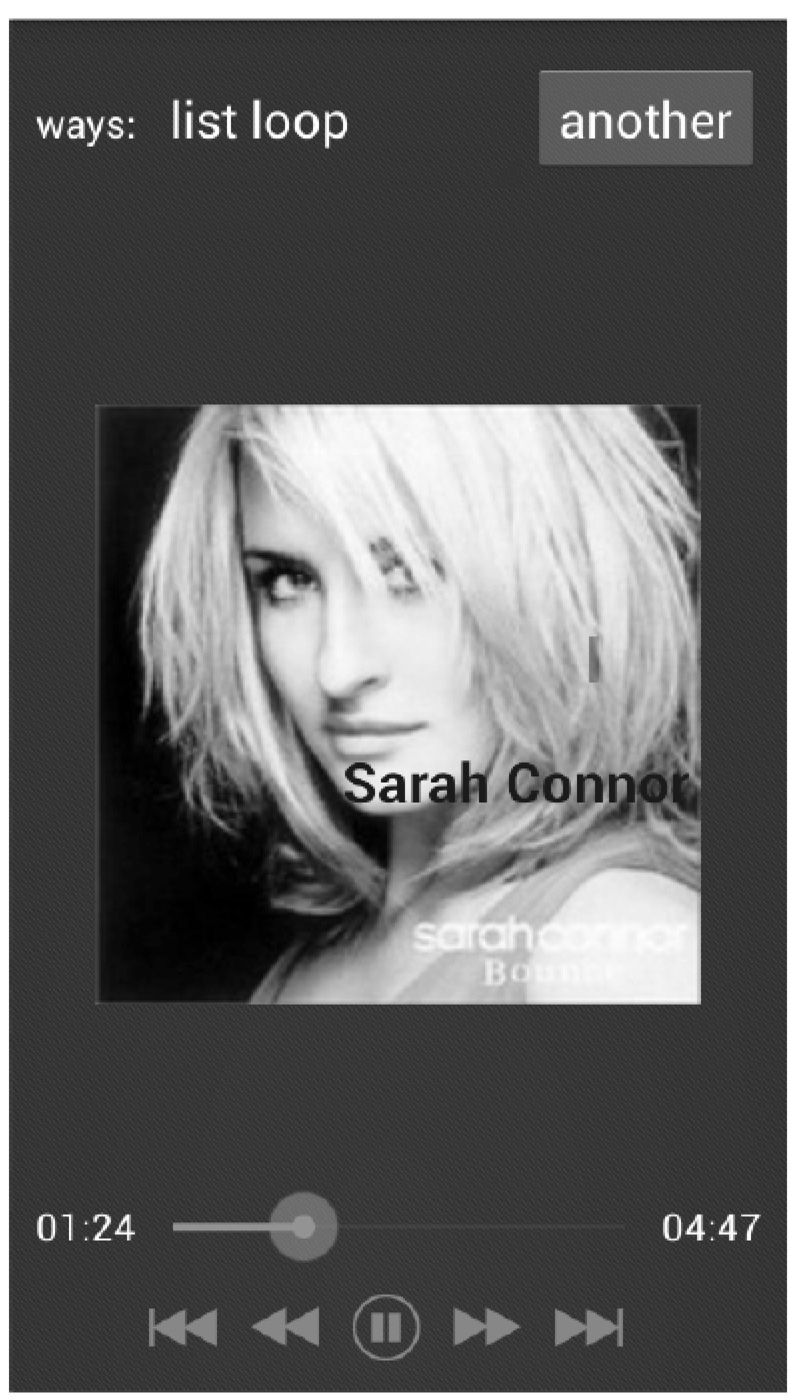


Figure 17-1 the figure of the running results with different operations









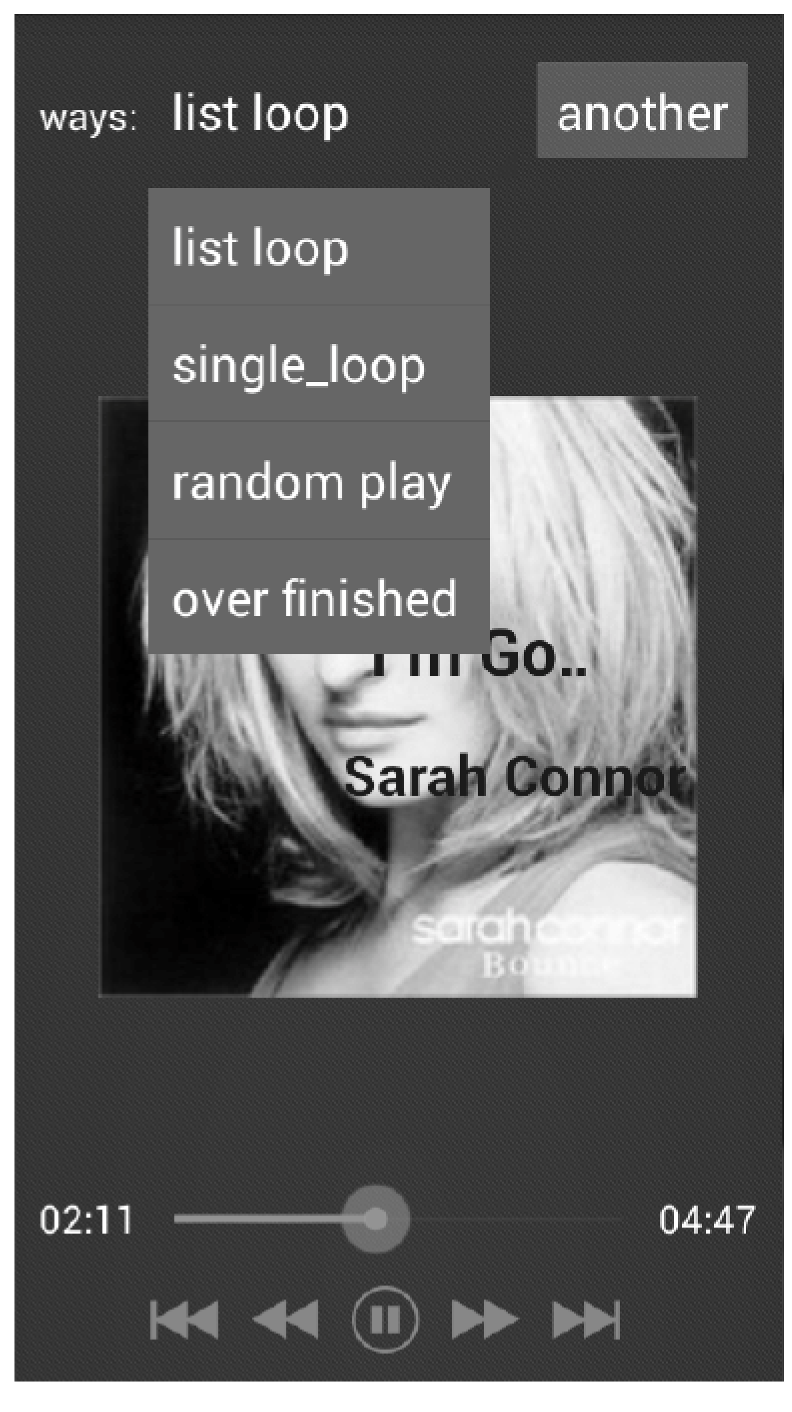




Figure 17-1(continued)

17.2 Key Code

Main interface layout file: 17\MusicPlayer\res\layout\activity_main.xml

```
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mTabHost"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
         <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:background="@drawable/content_bg">
            <FrameLayout
                android:id="@+id/realContent"
                android:layout_width="match_parent"
                android:layout_height="match_parent" />
        </FrameLayout>
    </LinearLayout>
</TabHost>
```

Single item layout file: 17\ MusicPlayer\res\layout\tab.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:padding="5dp"
    android:background="@drawable/bg"
    android:orientation="vertical">
    <ImageView
        android:id="@+id/icon"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:contentDescription="@string/imageInfo" />
    <TextView
        android:id="@+id/title"
        android:textColor="#ffffff"
        android:textStyle="bold"
        android:textSize="12sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

MainActivity: 17\MusicPlayer\src\iet\jxufe\cn\android\musicplayer\MainActivity.java

```
public class MainActivity extends Activity {
    private TabHost mTabHost;
    private String[] titles = new String[] {"artist", "music", "album",
"playlist"};
    private String[] tags = new String[] { "artist", "music", "album",
"playlist"};
    private int[] icons = new int[] { R.drawable.music,
R.drawable.artist,
            R.drawable.album, R.drawable.playlist };
    private MyOpenHelper mHelper;
    private SQLiteDatabase mDatabase;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);//remove the
                                                       //title Bar
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_main);
        initData();
        mTabHost = (TabHost) findViewById(R.id.mTabHost);
        mTabHost.setup();
        for (int i = 0; i < titles.length; i++) {//Add tabs circularly.
            TabSpec tabSpec = mTabHost.newTabSpec(tags[i]);
            //create an tab, and specify its tag
            View view = getLayoutInflater().inflate(R.layout.tab, null);
            TextView titleView = (TextView) view.findViewById(R.id.title);
            ImageView iconView = (ImageView) view.findViewById (R.id.icon);
            titleView.setText(titles[i]); //set title of the tab
            iconView.setImageResource(icons[i]); //set icon of the tab
            tabSpec.setIndicator(view);
            tabSpec.setContent(R.id.realContent); //set content for the
                                                  //tab
            mTabHost.addTab(tabSpec); //add tab into tabHost
        }
    mTabHost.setOnTabChangedListener(new     MyTabChangedListener());
   //add tab change listener for tabhost
        mTabHost.setCurrentTab(1);//display the second by default
    }
    public void initData(){
        mHelper=new MyOpenHelper(this, "music",null, 1);
        mDatabase=mHelper.getWritableDatabase();
        Constants.playlist=MusicUtils.getDataFromDB(mDatabase);
```

```
            //get musics from database
        }
        private class MyTabChangedListener implements OnTabChangeListener {
            public void onTabChanged(String tabTag) {
                FragmentTransaction fragmentTransaction =
                                                       getFragmentManager()
                        .beginTransaction();
                if (tabTag.equalsIgnoreCase("music")) {
                //switch to the music list tab
                    MusicListFragment musicListFragment=new
                    MusicListFragment();
                    musicListFragment.setMusicList(null);
                    fragmentTransaction.replace(R.id.realContent,
                            musicListFragment);
                } else if (tabTag.equalsIgnoreCase("artist")) {
                //switch to the artist classification tab
                    fragmentTransaction.replace(R.id.realContent,
                            new ArtistListFragment());
                } else if (tabTag.equalsIgnoreCase("album")) {
                //switch to the album classification tab
                    fragmentTransaction.replace(R.id.realContent,
                            new AlbumListFragment());
                } else if (tabTag.equalsIgnoreCase("playlist")) {
                    fragmentTransaction.replace(R.id.realContent,
                            new PlayListFragment());
                }
                fragmentTransaction.commit();
            }
        }
        protected void onDestroy() {//save the data in the playlist into
                                    //database when exiting
            mDatabase.execSQL("delete from music_tb");
             //delete all existing data
            for(int i=0;i<Constants.playlist.size();i++){
             //iterate to access the music in the list
                Music music=Constants.playlist.get(i); //get music
                mDatabase.execSQL("insert into music_tb(title,artist, album,
                    album_id,time,url)values (?,?,?,?,?,?)",new String[]
                    music.getTitle(),music.getSinger(),music.getAlbum(),
                   music.getAlbum_id()+"",music.getTime()+"",music.getUrl()});
                   //save the music information into database
            }
            super.onDestroy();
        }
    }
```

Music class: 17\ MusicPlayer\src\iet\jxufe\cn\android\musicplayer\Music.java

```
public class Music implements Serializable {
    private static final long serialVersionUID=1;
    private String title;//song title.
    private String singer;
    private String album;
    private int album_id;
    private String url;//song file path
    private long size;
    private int time;//the duration of song file
    private String name;
    public int getAlbum_id() {
        return album_id;
    }
    public void setAlbum_id(int album_id) {
        this.album_id = album_id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getSinger() {
        return singer;
    }
    public void setSinger(String singer) {
        this.singer = singer;
    }
    public String getAlbum() {
        return album;
    }
    public void setAlbum(String album) {
        this.album = album;
    }
    public String getUrl() {
        return url;
    }
    public void setUrl(String url) {
        this.url = url;
    }
    public long getSize() {
        return size;
    }
```

```
    public void setSize(long size) {
        this.size = size;
    }
    public int getTime() {
        return time;
    }
    public void setTime(int time) {
        this.time = time;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String toString() {//display song name and artist
        return "Music [title=" + title + ", singer=" + singer + "]";
    }
}
```

Music tool class: 17\MusicPlayer\src\iet\jxufe\cn\android\musicplayer\MusicUtils.java

```
public class MusicUtils {
    public static List<Music> getMusicData(Context context) {
        ContentResolver mResolver = context.getContentResolver();
        //get ContentResovler
        if (mResolver != null) {//get all the musics
            //the first parameter represents the URI of music provider
            //in the system
            //the second parameter represents the row information need
            //to get
            //the third parameter represents the query condition
            //the fourth parameter represents the value of placeholder
            //in the condition
            //the fifth parameter represents the sortord of query results
            Cursor cursor = mResolver.query(
                MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, null, null,
                null, MediaStore.Audio.Media.DEFAULT_SORT_ORDER);
            return cursorToList(cursor, context);
        }
        return null;
    }
    public static List<Music> cursorToList(Cursor cursor, Context
context){
        if (cursor == null || cursor.getCount() == 0) {
```

```
            return null;
        }
        List<Music> musicList = new ArrayList<Music>();
        while (cursor.moveToNext()) {
            Music m = new Music();
            String title = cursor.getString(cursor
                    .getColumnIndex(MediaStore.Audio.Media.TITLE));
            String artist = cursor.getString(cursor
                    .getColumnIndex(MediaStore.Audio.Media.ARTIST));
            String album = cursor.getString(cursor
                    .getColumnIndex(MediaStore.Audio.Media.ALBUM));
            int album_id = cursor.getInt(cursor
                   .getColumnIndex(MediaStore.Audio.Media.ALBUM_ID));
            long size = cursor.getLong(cursor
                    .getColumnIndex(MediaStore.Audio.Media.SIZE));
            int time = cursor.getInt(cursor
                   .getColumnIndex(MediaStore.Audio.Media.DURATION));
            String url = cursor.getString(cursor
                    .getColumnIndex(MediaStore.Audio.Media.DATA));
            String name = cursor.getString(cursor
                  .getColumnIndex(MediaStore.Audio.Media.DISPLAY_NAME));
            String sub = name.substring(name.lastIndexOf(".") + 1);
            //get the file extension name
            if (sub.equals("mp3") && time > 50000) {
            //end by MP3 and the duration greater than 5 seconds
                m.setTitle(title);
                m.setSinger(artist);
                m.setAlbum(album);
                m.setAlbum_id(album_id);
                m.setSize(size);
                m.setTime(time);
                m.setUrl(url);
                m.setName(name);
                musicList.add(m);
            }
        }
        cursor.close();//close the cursor
        return musicList;
    }
    public static String timeToString(int time) {//time format
        conversion, convert milliseconds to minutes and seconds
        int temp = time / 1000;//convert milliseconds to seconds.
        int minute = temp / 60;//calculate the total number of minutes.
        int second = temp % 60;//the rest seconds besides these minutes
```

```
        return String.format("%02d:%02d", minute, second);
        //display as minutes and seconds
    }
    public static List<Music> getDataFromDB(SQLiteDatabase db) {
        List<Music> musics = new ArrayList<Music>();
        Cursor cursor = db.rawQuery("select * from music_tb", null);
        if (cursor == null || cursor.getCount() == 0) {
            return musics;
        }
        while (cursor.moveToNext()) {
            Music music = new Music();
            music.setTitle(cursor.getString(cursor.getColumnIndex(("title"))));
            music.setSinger(cursor.getString(cursor.getColumnIndex("artist")));
            music.setAlbum(cursor.getString(cursor.getColumnIndex("album")));
            music.setAlbum_id(cursor.getInt(cursor.getColumnIndex("album_id")));
            music.setUrl(cursor.getString(cursor.getColumnIndex("url")));
            music.setTime(cursor.getInt(cursor.getColumnIndex("time")));
            musics.add(music);
        }
        return musics;
    }
    public static Bitmap getAlbumPic(Context context, Music music) {
        ContentResolver mResolver = context.getContentResolver();
        Uri uri = ContentUris.withAppendedId(Constants.ALBUM_URL, music.getAlbum_id());
        try {
            InputStream inputStream = mResolver.openInputStream(uri);
            return BitmapFactory.decodeStream(inputStream);
        } catch (FileNotFoundException ex) {
        //If not exist, throws exception
            try {
                ParcelFileDescriptor pfd = mResolver.openFileDescriptor(uri, "r");
                if (pfd != null) {
                    FileDescriptor fd = pfd.getFileDescriptor();
                    Bitmap bitmap = BitmapFactory.decodeFileDescriptor(fd);
                    return bitmap;
                }
```

```
            } catch (Exception e) {
                return null;
            }
            return null;
        }
    }
}
```

Database helper class: 17\ MusicPlayer\src\iet\jxufe\cn\android\musicplayer\MyOpenHelper.java

```
public class MyOpenHelper extends SQLiteOpenHelper {
    public String createTableSQL="create table if not exists music_tb" +
            "(_id integer primary key autoincrement, title, artist,
album, album_id,time,url)";
    public MyOpenHelper(Context context, String name, CursorFactory
factory,
            int version) {
        super(context, name, factory, version);
    }
    //method will be invoked when database is created, execute the
    //instruction of creating table and insert initialized data
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(createTableSQL);
    }
    //method will be invoked when the version is updated
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
    System.out.println("version changed:"+oldVersion+"-->"+ newVersion);
    }
}
```

Music list Fragment: 17\MusicPlayer\src\iet\jxufe\cn\android\musicplayer\MusicListFragment.java

```
public class MusicListFragment extends ListFragment {
    //the tab displays all music information by default, so we start
    //background music service here
    public List<Music> musicList;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        musicList = getMusicList();
        if (musicList == null || musicList.size() == 0) {
        //if the musicList is empty
            musicList = MusicUtils.getMusicData(getActivity());
            //get all the music
        }
    }
```

```
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        Constants.musiclist = musicList;
        if (musicList != null) {
            setListAdapter(new MusicAdapter());//show music list
        } else {
            Toast.makeText(getActivity(), "there aren't musics in
storage, please add some...",
        Toast.LENGTH_SHORT).show();
         //remind user that there is no music in SDCard
        }
        Intent intent = new Intent(getActivity(), MusicService.class);
        //create Intent, start the specific Service.
        getActivity().startService(intent);
        return super.onCreateView(inflater, container,
                                 savedInstanceState);
    }
    public void onStart() {
        registerForContextMenu(getListView());
         //add a context menu for the music listView
        super.onStart();
    }
    private class MusicAdapter extends BaseAdapter {
        public int getCount() {
            return musicList.size();
        }
        public Object getItem(int position) {
            return musicList.get(position);
        }
        public long getItemId(int position) {
            return position;
        }
        public View getView(int position, View convertView, ViewGroup
            parent) {
            if (convertView == null) {
                convertView = LinearLayout.inflate(getActivity(),
                        R.layout.music_item, null);
            }
            ImageView icon = (ImageView) convertView.findViewById
(R.id.icon);
            TextView title = (TextView) convertView.findViewById
(R.id.title);
            TextView artist = (TextView) convertView.findViewById
(R.id.artist);
            TextView time = (TextView) convertView.findViewById
```

```
(R.id.time);
            Bitmap bitmap = MusicUtils.getAlbumPic(getActivity(),
musicList.get(position));
            if(bitmap!=null){//if album picture is not empty, display
                it;if it is empty, then display the default picture.
                icon.setImageBitmap(bitmap);
            }else {
                icon.setImageResource(R.drawable.music);
                //display the default picture
            }
            title.setText(musicList.get(position).getTitle());
            artist.setText(musicList.get(position).getSinger());
            time.setText(MusicUtils.timeToString(musicList.get(position)
                    .getTime()));
            return convertView;
        }
    }
    public List<Music> getMusicList() {
        return musicList;
    }
    public void setMusicList(List<Music> musicList) {
        this.musicList = musicList;
    }
    public void onListItemClick(ListView l, View v, int position, long
         id) {//music items click event handler
        Intent intent = new Intent(getActivity(),
                                    MusicPlayActivity.class);
        intent.putExtra("listType", Constants.ALL_MUSIC);
         //music list type: list all the musics
        intent.putExtra("music", musicList.get(position));
         //current music
        intent.putExtra("position", position);
         //corresponding position to the current music
        startActivity(intent);
        super.onListItemClick(l, v, position, id);
    }
    public void onCreateContextMenu(ContextMenu menu, View v,
            ContextMenuInfo menuInfo) {//creating a context menu
        getActivity().getMenuInflater().inflate(R.menu.musiclist_
context, menu);
        super.onCreateContextMenu(menu, v, menuInfo);
    }
    public boolean onContextItemSelected(MenuItem item) {
        AdapterContextMenuInfo info = (AdapterContextMenuInfo) item
                .getMenuInfo();//get the context menu information
```

```
        switch (item.getItemId()) {
        case R.id.setToBell://set as ringtone
            setRing(musicList.get(info.position));
            break;
        case R.id.addToPlayList://add to favorites list.
            Music music = musicList.get(info.position);
            int i = 0;
            for (; i < Constants.playlist.size(); i++) {
            //check whether the playlist contains the music
                if (Constants.playlist.get(i).getTitle()
                        .equalsIgnoreCase(music.getTitle())) {
                    break;//if exists, exit directly.
                }
            }
            if (i == Constants.playlist.size()) {
                Constants.playlist.add(music);
            }
            break;
        default:
            break;
        }
        return super.onContextItemSelected(item);
    }
    public void setRing(Music music) {//set as ringtone
        ContentValues values = new ContentValues();
        values.put(MediaStore.MediaColumns.DATA, music.getUrl());
         //music path
        values.put(MediaStore.MediaColumns.MIME_TYPE, "audio/mp3");
        values.put(MediaStore.Audio.Media.IS_RINGTONE, true);
        //whether set it as ringtone
        values.put(MediaStore.Audio.Media.IS_NOTIFICATION, false);
        values.put(MediaStore.Audio.Media.IS_ALARM, false);
        values.put(MediaStore.Audio.Media.IS_MUSIC, false);
        Uri uri = MediaStore.Audio.Media.getContentUriForPath
(music.getUrl());//get corresponding URI according to the path.
        Uri newUri = getActivity().getContentResolver().insert(uri,
values);//insert a new value
        RingtoneManager.setActualDefaultRingtoneUri(getActivity(),
                                RingtoneManager.TYPE_RINGTONE, newUri);
    }
}
```

Music list Fragment layout file: 17\MusicPlayer\res\layout\music_item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/icon"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_margin="5dp"
        android:contentDescription="@string/imageInfo" />
    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical"
        android:padding="5dp" >
        <TextView
            android:id="@+id/title"
            android:singleLine="true"
            android:ellipsize="end"
            android:paddingRight="10dp"
            android:textSize="18sp"
            style="@style/textStyle"/>

        <TextView
            android:id="@+id/artist"
            style="@style/textStyle" />
    </LinearLayout>
    <TextView
        android:id="@+id/time"
        android:paddingRight="20dp"
        android:textSize="18sp"
        style="@style/textStyle" />
</LinearLayout>
```

Artist list Fragment: 17\MusicPlayer\src\iet\jxufe\cn\android\musicplayer\ArtistListFragment.java

```
public class ArtistListFragment extends ListFragment {
    private List<Music> musicList;
    private List<MusicGroupByArtist> artistsList = new ArrayList
<MusicGroupByArtist>();
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        musicList = MusicUtils.getMusicData(getActivity());
        if (musicList != null) {
```

```
                MusicGroupByArtist(musicList);//classify music
            } else {
                Toast.makeText(getActivity(), "there aren't musics in
storage, please add some...",
                        Toast.LENGTH_SHORT).show();
            }
        }
        public void MusicGroupByArtist(List<Music> musicList) {
            //classify music by artist, and staticitc the number of music
            //of each artist
            for (int i = 0; i < musicList.size(); i++) {
                //iterate to access each music, judge the album
                int j = 0;
                for (; j < artistsList.size(); j++) {//iterate to access each
                                    //artist, judge whether the artisit is exist
                    if (musicList.get(i).getSinger()
                        .equals(artistsList.get(j).getArtistName())) {
                        //if exists, add the music to the artist
                        artistsList.get(j).setCount(
                                artistsList.get(j).getCount() + 1);
                            //plus one for the number
                        artistsList.get(j).getMusics().add(musicList.get(i));
                        //add the music to the collection
                        break;//exit the iteration
                    }
                }
                if (j == artistsList.size()) {
                    //if the artist is not exist, create it
                    MusicGroupByArtist artist = new MusicGroupByArtist();
                    //create an artist
                    artist.setArtistName(musicList.get(i).getSinger());
                    //set the name of the artist
                    artist.setCount(1);//the default song number is 1
                    List<Music> musics = new ArrayList<Music>();
                    //create a collection for saving all the music of this
                    //artist
                    musics.add(musicList.get(i));
                    artist.setMusics(musics);
                    artistsList.add(artist);
                }
            }
        }
        private class MusicGroupByArtist {//Artist group information
            private String artistName;
            private int count;//the count of music.
```

```
        private List<Music> musics;
        //set and get method
        public String getArtistName() {
            return artistName;
        }
        public void setArtistName(String artistName) {
            this.artistName = artistName;
        }
        public int getCount() {
            return count;
        }
        public void setCount(int count) {
            this.count = count;
        }
        public List<Music> getMusics() {
            return musics;
        }
        public void setMusics(List<Music> musics) {
            this.musics = musics;
        }
    }
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        setListAdapter(new ArtistAdapter());
         //show results sorted by artist
        return super.onCreateView(inflater, container,
                                    savedInstanceState);
    }
    private class ArtistAdapter extends BaseAdapter {
        public int getCount() {
            return artistsList.size();
        }
        public Object getItem(int position) {
            return artistsList.get(position);
        }
        public long getItemId(int position) {
            return position;
        }
        public View getView(int position, View convertView, ViewGroup
                                parent) {
            if (convertView == null) {
                convertView = LinearLayout.inflate(getActivity(),
                        R.layout.artist_item, null);
            }
```

```
                ImageView icon = (ImageView)convertView.findViewById(R.id.icon);
                TextView album = (TextView) convertView.findViewById(R.id.artist);
                TextView info = (TextView) convertView.findViewById(R.id.info);
                Bitmap bitmap = MusicUtils.getAlbumPic(getActivity(),
                             artistsList.get(position).getMusics().get(0));
                if(bitmap!=null){
                     icon.setImageBitmap(bitmap);
                }else {
                     icon.setImageResource(R.drawable.artist);
                }
                album.setText(artistsList.get(position).getArtistName());
                if(artistsList.get(position).getCount()>1){
                     info.setText(Html.fromHtml("there are <font color=red>
                          <b>" +artistsList.get(position).getCount() + "</b>
                          </font> musics"));
                }else{
                info.setText(Html.fromHtml("there is <font color=red><b>" +
                     artistsList.get(position).getCount() + "</b></font>
                     music"));
                return convertView;
           }
      }
     public void onListItemClick(ListView l, View v, int position, long id) {          //display all the music under this artist after click.
           MusicListFragment musicListFragment=new MusicListFragment();
           musicListFragment.setMusicList(artistsList.get(position).getMusics());
           FragmentTransaction fTransaction=getActivity().getFragmentManager().beginTransaction();
           fTransaction.replace(R.id.realContent, musicListFragment);
           fTransaction.commit();
           super.onListItemClick(l, v, position, id);
      }
}
```

Artist list Fragment layout file: 17\MusicPlayer\res\layout\ MusicPlayer\res\layout\artist_item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical"
    android:orientation="horizontal" >
```

```
    <ImageView
        android:id="@+id/icon"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_margin="5dp"
        android:contentDescription="@string/imageInfo" />
    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical"
        android:padding="5dp" >
        <TextView
            android:id="@+id/artist"
            android:singleLine="true"
            android:ellipsize="end"
            android:paddingRight="10dp"
            android:textSize="18sp"
            style="@style/textStyle"/>
        <TextView
            android:id="@+id/info"
            style="@style/textStyle" />
    </LinearLayout>
</LinearLayout>
```

Album list Fragment: 17\ MusicPlayer\src\iet\jxufe\cn\android\musicplayer\AlbumListFragment.java

```
public class AlbumListFragment extends ListFragment {
    private List<Music> musicList;
    private List<MusicGroupByAlbum> albumList = new ArrayList
<AlbumListFragment.MusicGroupByAlbum>();
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        musicList = MusicUtils.getMusicData(getActivity());
        if (musicList != null) {
            MusicGroupByAlbum(musicList);
        } else {
            Toast.makeText(getActivity(), "there aren't musics in
storage, please add some...",
                    Toast.LENGTH_SHORT).show();
        }
    }
    public void MusicGroupByAlbum(List<Music> musicList) {
        //classify music by album and statistic and statistics the number
        //of music in each album
```

```
        for (int i = 0; i < musicList.size(); i++) {
         //iterate to access each music, get the album
            int j = 0;
            for (; j < albumList.size(); j++) {//iterate to access each
                                  //album, judge whether the album is existed
                if (musicList.get(i).getAlbum()
                        .equals(albumList.get(j).getAlbumName())) {
                        //if exists, add the music to the album
                    albumList.get(j).setCount(albumList.get(j).
                     getCount()+1);//plus one for the number
                    albumList.get(j).getMusics().add(musicList.get(i));
                    //add the music to the collection
                    break;//exit the iteration
                }
            }
            if (j == albumList.size()) {
                 //if the album is not exist, create it
                MusicGroupByAlbum album = new MusicGroupByAlbum();
               //create an album
                album.setAlbumName(musicList.get(i).getAlbum());
                album.setCount(1);//the default song number is 1
                List<Music> musics = new ArrayList<Music>();
                musics.add(musicList.get(i));
                album.setMusics(musics);
                albumList.add(album);
            }
        }
    }
    private class MusicGroupByAlbum {//grouped by album information
        private String albumName;
        private int count;
        private List<Music> musics;
        //set and get method.
        public String getAlbumName() {
            return albumName;
        }
        public void setAlbumName(String albumName) {
            this.albumName = albumName;
        }
        public int getCount() {
            return count;
        }
        public void setCount(int count) {
            this.count = count;
```

```
        }
        public List<Music> getMusics() {
            return musics;
        }
        public void setMusics(List<Music> musicList) {
            this.musics = musicList;
        }
    }
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        setListAdapter(new AlbumAdapter());//show all album information.
        return super.onCreateView(inflater, container, savedInstanceState);
    }
    private class AlbumAdapter extends BaseAdapter {
        public int getCount() {
            return albumList.size();
        }
        public Object getItem(int position) {
            return albumList.get(position);
        }
        public long getItemId(int position) {
            return position;
        }
        public View getView(int position, View convertView, ViewGroup parent) {
            if (convertView == null) {
                convertView = LinearLayout.inflate(getActivity(),
                        R.layout.album_item, null);
            }
            ImageView icon = (ImageView) convertView.findViewById(R.id.icon);
            TextView album = (TextView) convertView.findViewById(R.id.album);
            TextView info = (TextView) convertView.findViewById(R.id.info);
            Bitmap bitmap = MusicUtils.getAlbumPic(getActivity(), albumList
                    .get(position).getMusics().get(0));
            if (bitmap != null) {
                icon.setImageBitmap(bitmap);
            } else {
                icon.setImageResource(R.drawable.album);
            }
            album.setText(albumList.get(position).getAlbumName());
```

```
                if(albumList.get(position).getCount()>1){
                    info.setText(Html.fromHtml("there are <font color=red><b>"
                        +albumList.get(position).getCount() + "</b></font>
                         musics"));
                }else{
                    info.setText(Html.fromHtml("there is <font color = red>
                        <b>" + albumList.get(position).getCount() + "</b>
                         </font> music"));
                }
                return convertView;
            }
        }
        public void onListItemClick(ListView l, View v, int position, long id){
            //display all the music information in the album when it is
            //selected
            MusicListFragment musicListFragment = new MusicListFragment();
            musicListFragment.setMusicList(albumList.get(position).
getMusics());
            FragmentTransaction fTransaction = getActivity().
getFragmentManager()
                    .beginTransaction();
            fTransaction.replace(R.id.realContent, musicListFragment);
            fTransaction.commit();
            super.onListItemClick(l, v, position, id);
        }
}
```

Album list Fragment layout file: 17\ MusicPlayer\res\layout\album_item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/icon"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_margin="5dp"
        android:contentDescription="@string/imageInfo" />
    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical"
```

```
        android:padding="5dp" >
        <TextView
            android:id="@+id/album"
            android:singleLine="true"
            android:ellipsize="end"
            android:paddingRight="10dp"
            android:textSize="18sp"
            style="@style/textStyle"/>
        <TextView
            android:id="@+id/info"
            style="@style/textStyle" />
    </LinearLayout>
</LinearLayout>
```

Play list Fragment: 17\ MusicPlayer\src\iet\jxufe\cn\android\musicplayer\PlayListFragment.java

```
public class PlayListFragment extends ListFragment {
    public List<Music> musicList=Constants.playlist;
    private PlayListAdapter adapter;
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        if (musicList != null&&musicList.size()!=0) {
            adapter=new PlayListAdapter();
            setListAdapter(adapter);
        } else {
            Toast.makeText(getActivity(), "there aren't musics in storage, please add some...",
                    Toast.LENGTH_SHORT).show();
        }
        return super.onCreateView(inflater, container, savedInstanceState);
    }
    public void onStart() {
        registerForContextMenu(getListView());
        super.onStart();
    }
    private class PlayListAdapter extends BaseAdapter {
        public int getCount() {
            return musicList.size();
        }
        public Object getItem(int position) {
            return musicList.get(position);
        }
        public long getItemId(int position) {
            return position;
        }
```

```
        public View getView(int position, View convertView, ViewGroup parent) {
            if (convertView == null) {
                convertView = LinearLayout.inflate(getActivity(),
                        R.layout.music_item, null);
            }
            ImageView icon = (ImageView) convertView.findViewById(R.id.icon);
            TextView title = (TextView) convertView.findViewById(R.id.title);
            TextView artist = (TextView) convertView.findViewById(R.id.artist);
            TextView time = (TextView) convertView.findViewById(R.id.time);
            Bitmap bitmap=MusicUtils.getAlbumPic(getActivity(), musicList.get(position));
            if(bitmap!=null){
                icon.setImageBitmap(bitmap);
            }else {
                icon.setImageResource(R.drawable.music);
            }
            title.setText(musicList.get(position).getTitle());
            artist.setText(musicList.get(position).getSinger());
            time.setText(MusicUtils.timeToString(musicList.get(position)
                    .getTime()));
            return convertView;
        }
    }
    public void onListItemClick(ListView l,View v, int position, long id){
     //click music items event handler.
        Intent intent = new Intent(getActivity(), MusicPlayActivity.class);
        intent.putExtra("listType", Constants.PLAY_LIST_MUSIC);
        intent.putExtra("music", musicList.get(position));
        intent.putExtra("position", position);
         //the index of current music in the list
        startActivity(intent);
        super.onListItemClick(l, v, position, id);
    }
    public void onCreateContextMenu(ContextMenu menu, View v,
            ContextMenuInfo menuInfo) {
        getActivity().getMenuInflater().inflate(R.menu.playlist_context, menu);
        super.onCreateContextMenu(menu, v, menuInfo);
    }
```

```
    public boolean onContextItemSelected(MenuItem item) {
        AdapterContextMenuInfo info=(AdapterContextMenuInfo) item.
getMenuInfo();
        switch (item.getItemId()) {
        case R.id.deleteAll://Clear the playlist
            musicList.clear();
            System.out.println(musicList);
            break;
        case R.id.deleteFromList:
            musicList.remove(info.position);
            break;
        default:
            break;
        }
        adapter.notifyDataSetChanged();
        Constants.playlist=musicList;
        return super.onContextItemSelected(item);
    }
}
```

Constant class: 17\ MusicPlayer\src\iet\jxufe\cn\android\musicplayer\Constants.java

```
public class Constants {
    public static List<Music> musiclist=new ArrayList<Music>();
    public static List<Music> playlist=new ArrayList<Music>();
    public static final String CONTROL_ACTION = "iet.jxufe.cn.android.
       control";    //manage music play action, play or pause
    public static final String SEEKBAR_ACTION="iet.jxufe.cn.android.
       seekbar";    //play progress change action
    public static final String COMPLETE_ACTION="iet.jxufe.cn.android.
       complete";   //play over action
    public static final String UPDATE_ACTION="iet.jxufe.cn.android.
       update"; //update progress bar
    public static final String UPDATE_STYLE="iet.jxufe.cn.android.
       style";  //update play form
    public static final Uri ALBUM_URL=Uri.parse ("content://media/
       external/audio/albumart");
    public static final String LIST_LOOP="list loop";
    public static final String SINGLE_LOOP="single loop";
    public static final String OVER_FINISH="over finished";
    public static final String RANDOM_PLAY="random play";
    public static final int NEW=6;//start a new music.
    public static final int PLAY=1;//play
    public static final int PAUSE=2;//pause
    public static final int ALL_MUSIC=0x11;//play all musics
```

```
    public static final int PLAY_LIST_MUSIC=0x12; //play music in the list
}
```

Activity used to play music:
17\ MusicPlayer\src\iet\jxufe\cn\android\musicplayer\MusicPlayActivity.java

```
public class MusicPlayActivity extends Activity {//manage music playing
    private List<Music> musicList;
    private TextView titleView, singerView, currentTimeView, totalTimeView;
    private SeekBar playProgress;
    private Spinner styleSpinner;//spinner used for choosing play form
    private ImageButton control;//play/pause button
    private ImageView picView;
    private ServerReceiver serverReceiver;//broadcast Receiver used for
                                  //receiving broadcast send by background service
    private Music currentMusic;//current music
    private boolean isPause=false;//pause or no
    private int currentPosition;//the index of current music
    private int listType;//muisc list type: All the music or the music
                                  //in the play list
    private String[] styles=new String[] {Constants.LIST_LOOP, Constants.SINGLE_LOOP, Constants.RANDOM_PLAY,Constants.OVER_FINISH};
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE); //remove the title
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN); //full screen display
        setContentView(R.layout.play_item);//loading main interface.
        initView();
    }
    public void initView() {
        styleSpinner=(Spinner)findViewById(R.id.styleSpinner);
        styleSpinner.setAdapter(new ArrayAdapter<String> (this, R.layout.spinner_text,styles));
        styleSpinner.setOnItemSelectedListener(new SpinnerItemClickListener());
        PicView=(ImageView)findViewById(R.id.picView);
        currentTimeView = (TextView) findViewById(R.id.currentTime);
        totalTimeView = (TextView) findViewById(R.id.totalTime);
        titleView = (TextView) findViewById(R.id.title);
        singerView = (TextView) findViewById(R.id.singer);
```

```
        playProgress = (SeekBar) findViewById(R.id.playProgress);
        control = (ImageButton) findViewById(R.id.control);
        playProgress.setOnSeekBarChangeListener(new
MySeekBarChangeListener());
        //add event handler for the seekbar
        serverReceiver = new ServerReceiver();
         //create Broadcast Receiver
        IntentFilter filter = new IntentFilter();
         //broadcast type which can be received
        filter.addAction(Constants.COMPLETE_ACTION);
         //event of playing over
        filter.addAction(Constants.UPDATE_ACTION);
         //action of updating progress
        registerReceiver(serverReceiver, filter);
         //register Broadcast Receiver
        currentMusic=(Music) getIntent().getSerializableExtra("music");
         //get the current playing music
        if(currentMusic==null){
            SharedPreferences musicPreferences=getSharedPreferences
                                        ("music", Context.MODE_PRIVATE);
            currentPosition=musicPreferences.getInt("position", 0);
            listType=musicPreferences.getInt("listType",
Constants.ALL_MUSIC);
            if(listType==Constants.ALL_MUSIC){
                musicList=Constants.musiclist;
            }else{
                musicList=Constants.playlist;
            }
            currentMusic=musicList.get(currentPosition);
            //get the current music
            String styleString = musicPreferences.getString("style",
 Constants.LIST_LOOP);
            for(int i=0;i<styles.length;i++){
                if(styles[i].equalsIgnoreCase(styleString)){
                    styleSpinner.setSelection(i);
                    break;
                }
            }
            showInfo();//display music information
            isPause=true;//continue to play
            control(null);
        }else{//play music directly
            currentPosition=getIntent().getIntExtra("position",0);
            //play the first song by default
            listType=getIntent().getIntExtra("listType",
```

```
                Constants.ALL_MUSIC); //get the play type
                playNewMusic();
                if(listType==Constants.ALL_MUSIC){
                    musicList=Constants.musiclist;
                }else{
                    musicList=Constants.playlist;
                }
            }
        }
        public void showInfo(){
        totalTimeView.setText(MusicUtils.timeToString(currentMusic.
         getTime()));//show the total duration time of music
            titleView.setText("   " + currentMusic.getTitle() + "    ");
             //show the music title
            singerView.setText(currentMusic.getSinger());//show the singer
            Bitmap bitmap=MusicUtils.getAlbumPic(this, currentMusic);
            if(bitmap!=null){
                picView.setImageBitmap(bitmap);
            }else{
                picView.setImageResource(R.drawable.background);
            }
        }
        public void playNewMusic() {
            currentTimeView.setText(MusicUtils.timeToString(0));
             //show the current play time, default value is 0
            showInfo();
            playProgress.setProgress(0);//set progress as 0
            control.setImageResource(R.drawable.pause);
             //display the pause button
            //send a broadcast to the background to play music
            Intent controlIntent = new Intent(Constants.CONTROL_ACTION);
            controlIntent.putExtra("new", Constants.NEW);//a new music
            controlIntent.putExtra("position",currentPosition);
             //pass the serial number of current music
            controlIntent.putExtra("listType",listType);
             //pass the type of music play
            sendBroadcast(controlIntent);
            isPause=false;//set isPauset as false
        }
        private class MySeekBarChangeListener implements
    OnSeekBarChangeListener {
            public void onProgressChanged(SeekBar seekBar, int progress,
                    boolean fromUser) {
                   //call this method when the progress is changed
```

```
        }
        public void onStartTrackingTouch(SeekBar seekBar) {
         //call this method when start dragging
        }
        public void onStopTrackingTouch(SeekBar seekBar) {
         //call this method when end dragging
            Intent seekIntent = new Intent(Constants.SEEKBAR_ACTION);
            //send broadcast to inform the seekbar is changed
            seekIntent.putExtra("progress", seekBar.getProgress());
            //transfer the current progress
            sendBroadcast(seekIntent);
        }
    }
    private class ServerReceiver extends BroadcastReceiver {
        //broadcast Receiver used for receiving broadcast send by
        //background service
        public void onReceive(Context context, Intent intent) {
            if(intent.getAction()==Constants.UPDATE_ACTION){
            //broadcast of updating progress
                int position=intent.getIntExtra("position",0);
                 //get the playing position of music
                currentTimeView.setText(MusicUtils.timeToString
                (position)); //show the current playing time
                playProgress.setProgress((int)(position*1.0/
                 currentMusic.getTime()*100));
                //calculate the progress of progress bar according to
               //the position
            }else if(intent.getAction()==Constants.COMPLETE_ACTION){
                //Playing end event handler
                currentPosition=intent.getIntExtra("position", 0);
                 //get the serial number of current music
                currentMusic=musicList.get(currentPosition);
                showInfo();
            }
        }
    }
    private class SpinnerItemClickListener implements
         OnItemSelectedListener{
        public void onItemSelected(AdapterView<?> parent, View view,
                int position, long id) {
            Intent styleIntent=new Intent(Constants.UPDATE_STYLE);
            styleIntent.putExtra("style",styles[position]);
            sendBroadcast(styleIntent);
        }
        public void onNothingSelected(AdapterView<?> parent) {
```

```
            }
        }
        public void chooseMusic(View view){//handler of selecting song event
            Intent intent=new Intent(this,MainActivity.class);
            startActivity(intent);
            this.finish();
        }
        public void first(View view) {
         //handler of the first button click event
            currentPosition=0;
            currentMusic=musicList.get(currentPosition);
            playNewMusic();
        }
        public void pre(View view) {
        //handler of the previous button click event
        currentPosition=(currentPosition-1+musicList.size())%
         musicList.size();
            currentMusic=musicList.get(currentPosition);
            playNewMusic();
        }
        public void control(View view) {
         //handler of play and pause button click event
            Intent controlIntent=new Intent(Constants.CONTROL_ACTION);
             //manage the playing state of music, play or pause.
            if(!isPause){//if playing, send a broadcast to inform pause.
                controlIntent.putExtra("control",Constants.PAUSE);
                control.setImageResource(R.drawable.play);//change the icon
            }else{//if pause, send a broadcast to inform play
                controlIntent.putExtra("control",Constants.PLAY);
                control.setImageResource(R.drawable.pause);//Change the icon
            }
            isPause=!isPause;
            sendBroadcast(controlIntent);//send a broadcast
        }
        public void next(View view) {//handler of the next button click event
            currentPosition=(currentPosition+1)%musicList.size();
            currentMusic=musicList.get(currentPosition);
            playNewMusic();
        }
        public void last(View view) {//handler of the last button click event
            currentPosition=musicList.size()-1;
            currentMusic=musicList.get(currentPosition);
            playNewMusic();
        }
```

```
    protected void onDestroy() {//abolish the registration of Broadcast
                                //Receiver when the service is destroyed
        if (serverReceiver != null) {
            unregisterReceiver(serverReceiver);//Abolish the registration
when destroying the Activity.
        }
        super.onDestroy();
    }
}
```

Background Service: 17\ MusicPlayer\src\iet\jxufe\cn\android\musicplayer\MusicService.java

```
public class MusicService extends Service {
    private List<Music> musicList;
    private int position;
    private MediaPlayer mediaPlayer;
    private ActivityReceiver activityReceiver;
    private Music currentMusic;
    private Timer timer;
    private int listType;
    private String styleString=Constants.LIST_LOOP;
     //music play type, the default type is loop the list.
    public void onCreate() {//start service.
        mediaPlayer=new MediaPlayer();
        activityReceiver=new ActivityReceiver();
        IntentFilter filter=new IntentFilter();
        filter.addAction(Constants.CONTROL_ACTION);
        filter.addAction(Constants.SEEKBAR_ACTION);
        filter.addAction(Constants.UPDATE_STYLE);
        registerReceiver(activityReceiver, filter);
        super.onCreate();
    }
    public IBinder onBind(Intent intent) {
        return null;
    }
    private class ActivityReceiver extends BroadcastReceiver{
     //get the broadcast send by foreground.
        public void onReceive(Context context, Intent intent) {
            if(intent.getAction()==Constants.CONTROL_ACTION){
                //receive the broadcast of managing playing.
                int isNew=intent.getIntExtra("new", -1);
                if(isNew!=-1){//play a new music.
        listType=intent.getIntExtra("listType",Constants.ALL_MUSIC);
                    if(listType==Constants.ALL_MUSIC){
```

```
                    musicList=Constants.musiclist;
                }else{
                    musicList=Constants.playlist;
                }
                position=intent.getIntExtra("position", 0);
                currentMusic=musicList.get(position);
                 //get the current music which is ready to play
                preparedAndPlay(currentMusic);
                 //prepare and ready to play the music
            }else{
                int control=intent.getIntExtra("control",-1);
                if(control==Constants.PAUSE){
                    mediaPlayer.pause();//pause the music
                    timer.cancel();//cancel the timer
                }else if(control==Constants.PLAY){//continue playing
                    mediaPlayer.start();
                    startTimer();//start the timer
                }
            }
        }else if(intent.getAction()==Constants.SEEKBAR_ACTION){
            int progress=intent.getIntExtra("progress",0);
           //get the transferred progress
            int position=(int) (currentMusic.getTime()* progress*
1.0/100); //convert the the progress to corresponding time position
            mediaPlayer.seekTo(position);
           //skip to the specificed postion and keep on playing
        }else if(intent.getAction()==Constants.UPDATE_STYLE){
            styleString=intent.getStringExtra("style");
            SharedPreferences musicPreferences =
            getSharedPreferences ("music", Context.MODE_PRIVATE);
            Editor editor=musicPreferences.edit();
           //get the parameter editor
            editor.putString("style", styleString);
            editor.commit();
        }
    }
}
public void preparedAndPlay(Music music){
 //prepare and play the music
    try{
        mediaPlayer.reset();//reset the media player
        mediaPlayer.setDataSource(music.getUrl());
        //set the path of music
        mediaPlayer.prepare();
        mediaPlayer.start();
```

```
                startTimer();
                sendNotification();//send a broadcast
                saveInfo();//save the data
                mediaPlayer.setOnCompletionListener(new
                OnCompletionListener() {
                //the listener to listen playing over event
                    public void onCompletion(MediaPlayer mp) {
                    //When the playing is over, keep on playing according
                      to the setting and notice the foreground to change.
                        if(!Constants.OVER_FINISH.equalsIgnoreCase
(styleString)){
                            if(Constants.LIST_LOOP.equalsIgnoreCase
(styleString)){
                                position=(position+1)%musicList.size();
                                 //play next one automatically
                            }else  if(Constants.RANDOM_PLAY.equalsIgnoreCase
(styleString)){
                                position=new Random().nextInt(musicList.
                                   size());
                            }
                            currentMusic=musicList.get(position);
                            preparedAndPlay(currentMusic);
                            Intent intent=new Intent(Constants.COMPLETE_
                                                     ACTION);
                            intent.putExtra("position", position);
                            sendBroadcast(intent);
                        }else{
                            stopSelf();
                        }
                    }
                });
            }catch(Exception ex){
                ex.printStackTrace();
            }
        }
        public void saveInfo(){
            SharedPreferences musicPreferences=getSharedPreferences
                                     ("music", Context.MODE_PRIVATE);
            Editor editor=musicPreferences.edit();
            editor.putInt("listType",listType);//save the type of music list
            editor.putInt("position",position);//save the position of music
            editor.commit();
        }
        public void sendNotification(){//send notification in background
            NotificationManager notificationManager=(NotificationManager)
```

```
            getSystemService(Service.NOTIFICATION_SERVICE);
        Builder builder=new Notification.Builder(this);
        builder.setAutoCancel(false);
         //set AutoCancel as false when open the notice
        builder.setTicker("Music is playing");
         //prompt information displayed in the state bar
        builder.setSmallIcon(R.drawable.music);
         //set the small icon of the notification
    builder.setLargeIcon(BitmapFactory.decodeResource(getResources(),
     R.drawable.largeicon)); //set the large icon of the notification
        builder.setContentTitle("playing music");
        builder.setContentText(currentMusic.getTitle()+"
"+currentMusic.getSinger());//set the content of the notification
        Intent intent=new Intent("iet.jxufe.cn.android.music_play");
         //the page when the notification is started.
        PendingIntent pIntent=PendingIntent.getActivity(this,0,
intent, 0);
        builder.setContentIntent(pIntent);
         //set the program started by the notification
        notificationManager.notify(0x11, builder.build());
    }
    public void startTimer(){//start timer
        timer=new Timer();
        timer.schedule(new TimerTask() {
         //execute the assignment at regular time
            public void run() {//send broadcast to notice foreground to
                                  //update the progress bar
                Intent updateIntent=new Intent(Constants.UPDATE_ACTION);
                updateIntent.putExtra("position", mediaPlayer.
                                       getCurrentPosition());
                sendBroadcast(updateIntent);
            }
        }, 0,1000);
    }
    public void onDestroy() {//call this method when destroying Service
        if(mediaPlayer!=null){//reset the music player
            mediaPlayer.reset();
        }
        if(activityReceiver!=null){//abolish the Broadcast Receiver
            unregisterReceiver(activityReceiver);
        }
        super.onDestroy();
    }
}
```

Style file: 17\ MusicPlayer\res\values\styles.xml

```
<resources>
    <style name="AppBaseTheme" parent="android:Theme.Light">
    </style>
    <style name="AppTheme" parent="AppBaseTheme">
    </style>
     <style name="textStyle">
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#ffffff</item>
        <item name="android:textSize">14sp</item>
     </style>
     <style name="imageBtnStyle">
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:background">#00000000</item>
        <item name="android:layout_marginRight">10dp</item>
     </style>
</resources>
```

Manifest: 17\ MusicPlayer\AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="iet.jxufe.cn.android.musicplayer"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="17" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="iet.jxufe.cn.android.musicplayer.MainActivity"
            android:label="@string/app_name"
            android:launchMode="singleInstance">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
        <activity android:name="iet.jxufe.cn.android.musicplayer.MusicPlayActivity">
            <intent-filter>
                <action android:name="iet.jxufe.cn.android.music_play" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <service android:name="iet.jxufe.cn.android.musicplayer.MusicService"></service>
    </application>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!—Add this permission before testing in the real machine, if not, the application will return error:  refusing to reopen boot dex '/system/framework/hwframework.jar' -->
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
</manifest>
```

17.3 Code Analysis

17.3.1 Main functions of music player

This case implements some common features of a music player, including:

- Get all music of mp3 format in the memory card, displaying them in the form of list.
- Classify music like classify by the artist and album and make simple statistics, count the number of each type music. After click an item, show all music under this category.
- Realize playlist feature, users can add their favorite songs into the playlist and start playing music from it.
- Set a song as ringtone.
- Display the playing time and progress of current music, support change the music playback progress through the seekBar.
- Able to play music, pause, switch to the first song, previous song, next song and the last song.
- Supports a variety of play forms such as list cycle, single cycle, random play, stop when the play is finished and so on.

The main page jump and functional processes of music player instance are shown below

in Figure 17-2:

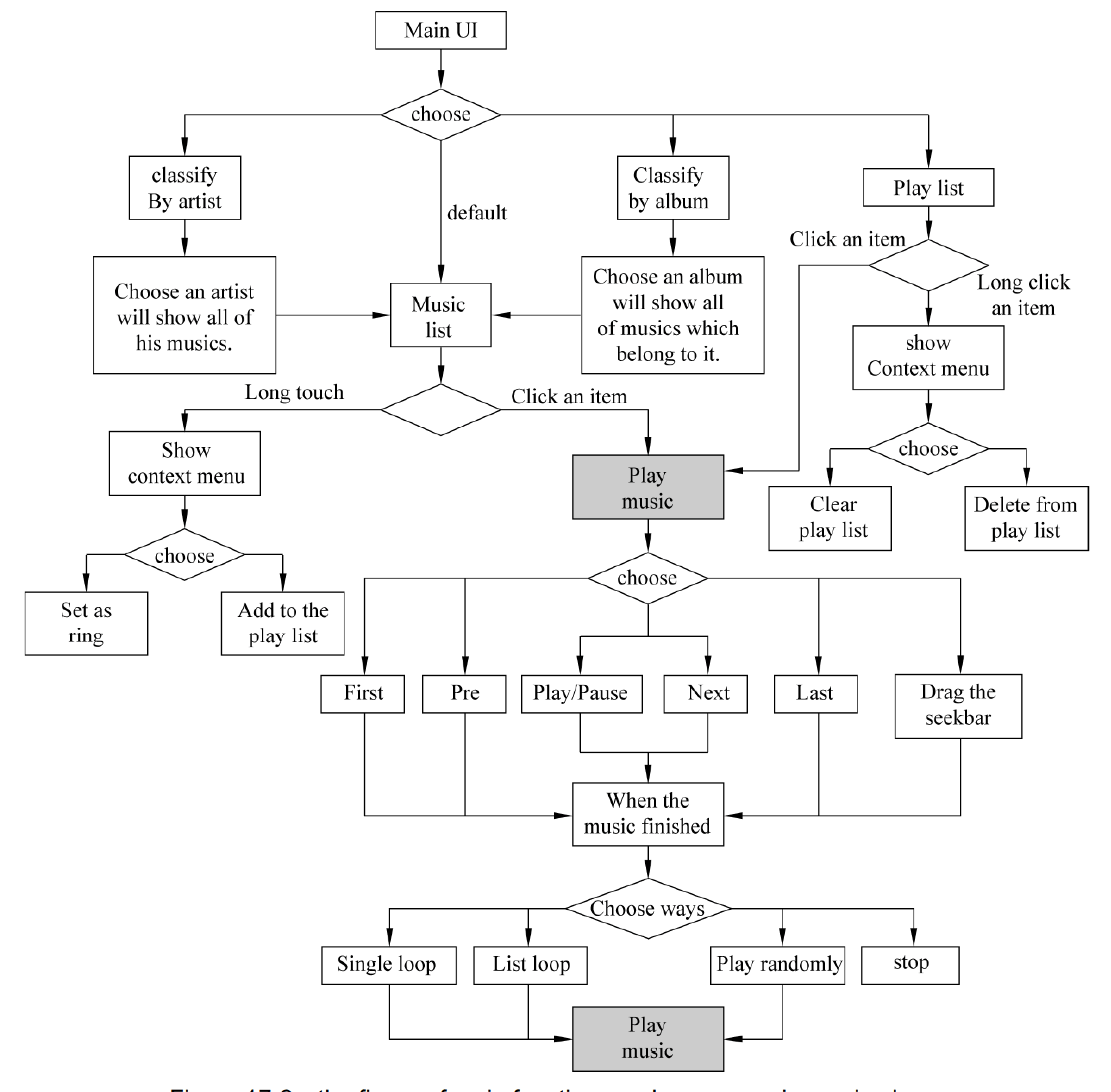


Figure 17-2 the figure of main functions and processes in music player

Switching playlist according to classification by artist, music list and album is primarily realized by TabHost + Fragment, there is only one Activity overall—MainActivity. We can call the ContentProvider which is provided by system to get all the music information in the memory card, then do some classifications and statistics.

Add a click event handler and context menu for music list, after one-click, jump to the play music page; pop-up the context menu after a long-press. The music can be set as ring tone or add to the playlist. Before add to the playlist, check whether the music has already exist in the playlist, if exist, give up adding.

In order to save the user's playlists without re-add every time, the application should store the playlist information in the local file when the user exit the program. When start again, get

related information from local file. As the music information is a structured data, this case stores music information by SQLite database.

17.3.2 ContentProvider

ContentProvider manage access to a structured set of data. They encapsulate the data, and provide mechanisms for defining data security. Content providers are the standard interface that connects data in one process with code running in another process. Android itself includes content providers that manage data such as audio, video, images, and personal contact information.

How does ContentProvider work ? A content URI is a URI that identifies data in a provider. Content URI include the symbolic name of the entire provider (its authority) and a name that points to a table (a path). URI is Uniform Resource Identifier, which means that each ContentProvider has its own unique URI. The ContentResolver object parses out the URI's authority, and uses it to "resolve" the provider by comparing the authority to a system table of known providers. The ContentResolver can then dispatch the query arguments to the correct provider. An application accesses the data from a content provider with a ContentResolver client object. This object has methods that call identically-named methods in the provider object, an instance of one of the concrete subclasses of ContentProvider. The ContentResolver methods provide the basic "CRUD" (create, retrieve, update, and delete) functions of persistent storage.

Content providers are one of the primary building blocks of Android applications. You implement a provider as one or more classes in an Android application, along with elements in the manifest file. One of your classes implements a subclass ContentProvider, which is the interface between your provider and other applications. Common method of ContentProvider base class :

- **public abstract boolean onCreate()**: Initialize your provider. The Android system calls this method immediately after it creates your provider. Notice that your provider is not created until a ContentResolver object tries to access it.
- **public abstract Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)**: Retrieve data from your provider. Use the arguments to select the table to query, the rows and columns to return, and the sort order of the result. Return the data as a Cursor object.
- **public abstract int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)**: Update existing rows in your provider. Use the arguments to select the table and rows to update and to get the updated column values. Return the number of rows updated.
- **public abstract int delete(Uri uri, String selection, String[] selectionArgs)**: Delete rows from your provider. Use the arguments to select the table and the rows to delete.

Return the number of rows deleted.

- **public abstract Uri insert(Uri uri, ContentValues values)**: Insert a new row into your provider. Use the arguments to select the destination table and to get the column values to use. Return a content URI for the newly-inserted row.
- **public abstract String getType(Uri uri)**: Return the MIME type corresponding to a content URI. This method is described in more detail in the section Implementing Content Provider MIME Types.

These methods are abstract methods that you must implement as part of your own concrete subclass. Like Activity components, a subclass of ContentProvider must be defined in the manifest file for its application, using the <provider> element. The symbolic name that identify the entire provider within the system is authorities, so we just need add some code in <application> element:

```
<provider android:name=".MyProvider"
    android:authorities="iet.jxufe.cn.android.provider.myprovider">
</provider>
```

Note: authorities is an essential attribute if no authorities the program will return error.

Once the application expose its data operation interface by ContentProvider, other applications can handle the inner data through that interface no matter whether the application starts or not.

There are three steps to operate the data exposed by ContentProvider in other applications:

(1) Get the URI of the application which provides data.

(2) Get ContentResolver object by call method getContentResolver() of current Context object.

(3) Call the CRUD methods of ContentResolver object to operate the exposed data.

We call the ContentProvider which is provided for audio and video and offered by system to get audio and video information in the device. The URI offered by system to access Sdcard is : MediaStore.Audio.Media.EXTERNAL_CONTENT_URI. The codes of getting music see line 2th-59th in MusicUtils.java. The URI to access album picture is : content://media/external/audio/albumart.Specific operation see: from 84th-103th line in MusicUtils.java. Set ring tone is also done by ContentProvider from 119th-130th line in MusicListFragment.java.

17.3.3 Service

A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service might handle network

transactions, play music, perform file I/O, or interact with a content provider, all from the background.

Service itself cannot run directly, it need the help of Context object. A service can essentially take two forms:

- Started: A service is “started” when an application component (such as an activity) starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed. Usually, a started service performs a single operation and does not return a result to the caller. For example, it might download or upload a file over the network. When the operation is done, the service should stop itself.
- Bound: A service is “bound” when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

Your service can work both ways—it can be started (to run indefinitely) and also allow binding. At this point, we need to call stopService() and unbindservice() together to destroy the Service.

Caution: A service runs in the main thread of its hosting process—the service does not create its own thread and does not run in a separate process (unless you specify otherwise). This means that, if your service is going to do any CPU intensive work or blocking operations (such as MP3 playback or networking), you should create a new thread within the service to do that work. By using a separate thread, you will reduce the risk of Application Not Responding (ANR) errors and the application's main thread can remain dedicated to user interaction with your activities.

Similar to Activity, you must create a subclass of Service to create a Service and override some callback methods. Main methods in Service:

- **public abstract IBinder onBind(Intent intent)**: Return the communication channel to the service. May return null if clients can not bind to the service. The returned IBinder is usually for a complex interface that has been described using IBinder.
- **public void onCreate()**: Called by the system when the service is first created. Do not call this method directly.
- **public void onDestroy()**: Called by the system to notify a Service that it is no longer used and is being removed.
- **public int onStartCommand(Intent intent, int flags, int startId)**: Called by the system every time a client explicitly starts the service by calling startService(Intent), providing the arguments it supplied and a unique integer token representing the start

request. Do not call this method directly.

- **public boolean onUnbind(Intent intent)**: Called when all clients have disconnected from a particular interface published by the service.

The custom subclass of Service must override method onBind(), in this case, class MusicService has override method onCreate()(Perform the initialization operation in this method) and onDestroy(Perform some outstanding work in this method) in addition. Then declare Service in the application's manifest file. You can specify which intent can start the Service by using element <intent-filter/> when configuring. The Service in this case is started explicitly, not conditionally, so we simply specify the Service's full class name here.

```
<service  android:name="iet.jxufe.cn.android.musicplayer.MusicService">
</service>
```

In this case, when the application is exited, we hope the music can still be played, so we start Service by method startService(). After the music plays, there is no relationship between the Service and Context which starts the Service. It comes a problem that how to pass the message to the backstage when we need to control music playback through foreground. How we communicate between foreground and backstage. In order to solve this problem, Android provides us another mechanism—BroadCast.

17.3.4 BroadcastReceiver

Broadcast is a widely used mechanism to transfer information between applications, the BroadcastReceiver is a component used to filter, receiver and respond broadcast. BroadcastReceiver is essentially a global listener used to monitor global system broadcast. Therefore, it can communicate with different components in the system easily.

Broadcast receivers enable applications to receive intents that are broadcast by the system or by other applications, even when other components of the application are not running. You can deliver a broadcast to other apps by passing an Intent to sendBroadcast(), sendOrderedBroadcast(), or sendStickyBroadcast(). Usually the intent can be received by several broadcast receivers which have subscribed to it. Like a broadcast station, can be received by many audiences.

BroadcastReceiver itself does not implement a graphical interface, but when it receives a message, it can start an Activity as respond, or remind the user via NotificationManager, or start a Service and so on. Start a BroadcastReceiver is very similar to start an Activity or Service, it needs two steps:

- Create the needed intent to start BroadcastReceiver.
- Call method sendBroadcast() or sendOrderedBroadcast() of Context object to start the specific BroadcastReceiver.

Develop a custom BroadcastReceiver is the same as develop other components in Android.

Create a subclass of base class BroadcastReceiver, implements related methods. Usually, we only need to implements the abstract method onReceive (Context context, Intent intent). After receiving the broadcast, this method will be callback, you can easily access some data transimt by broadcast through the incoming intent. Similar to other components, we need to register BroadcastReceiver after creating, but unlike other components, there are two ways to register it : One is to declare it in the manifest file with this element. The other is to create the receiver dynamically in code and register it with the Context.registerReceiver() method. Two ways are as follows:

Declare in the manifest file:

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter >
        <action 
android:name="iet.jxufe.cn.android.myBroadcastReceiver"></action>
    </intent-filter>
</receiver>
```

Registe in code dynamically:

```
MyBroadcastReceiver myBroadcastReceiver=new MyBroadcastReceiver();
IntentFilter filter=new IntentFilter();
filter.addAction("iet.jxufe.cn.android.myBroadcastReceiver");
registerReceiver(myBroadcastReceiver, filter);
```

The action represents the action which can be received by the broadcast receiver, a broadcast receiver can receiver many broadcast so that it can own many action attributes like the backstage service. It can either receive broadcast of play/pause or broadcast of change the music playback progress. Code see from 12th-17th line in MusicService.java. In this case, broadcast receivers are registered dynamically in code. In order to make the specified action when sending broadcast that same with the action which the broadcast receiver needs to avoid spelling mistake, all the actions in this case are separate defined as constants. Use the class name and constant name when you need to reference. See file Constants.java.

After registration, the application can receive corresponding broadcast. Once the broadcast event is happened, the system will create corresponding broadcast receiver object and trigger method onReceive() automatically. The BroadcastReceiver object will be destroyed after the method onReceive() is executed. The broadcast receiver can receive several broadcast in this case, so we need to judge the category of received broadcast through the action of received broadcast. The broadcast transfer relationship between Activity and Service is as follow in Figure 17-3.

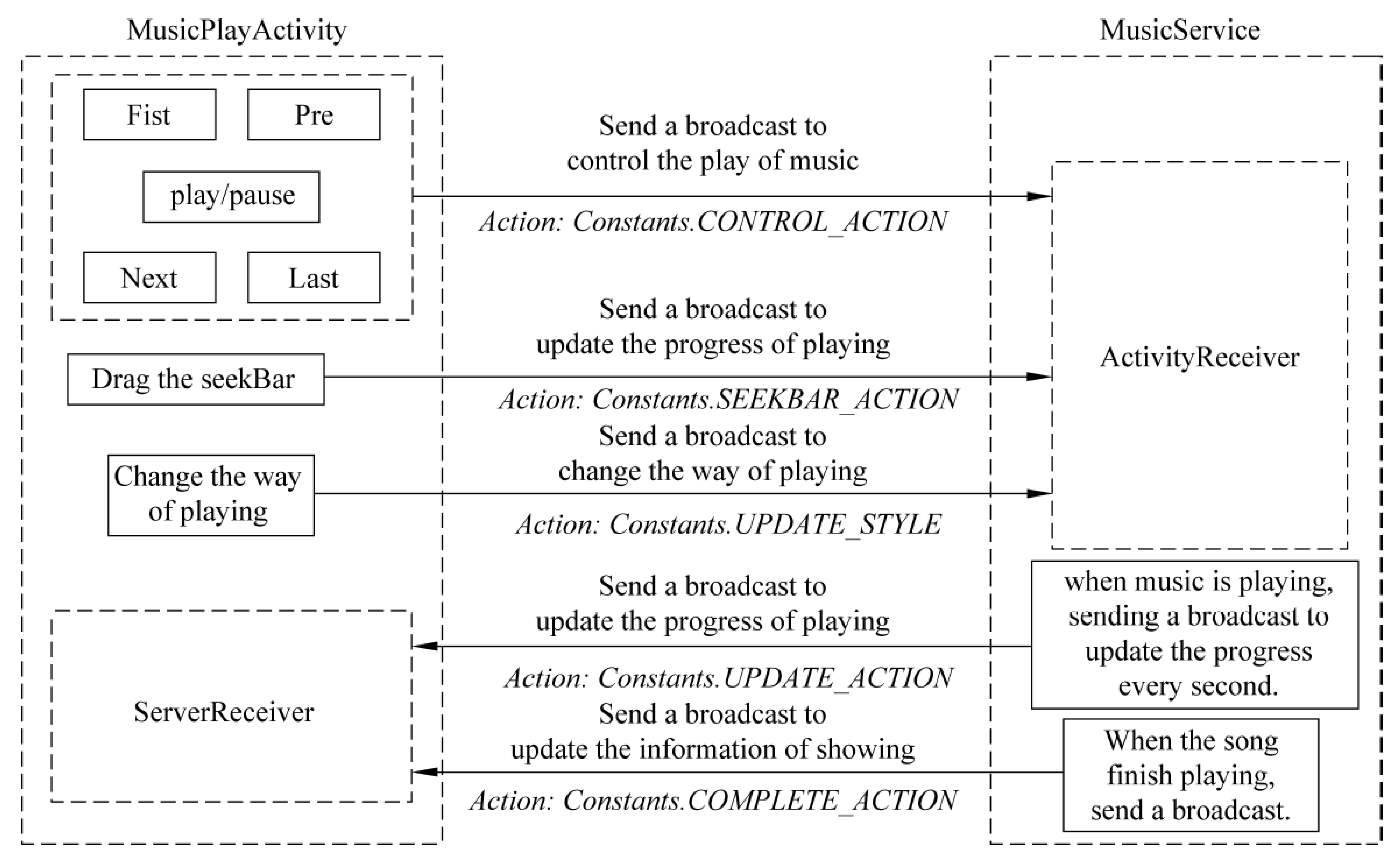


Figure 17-3 the figure of broadcast transmission between Activity and Service

17.4 Extension of Knowledge

17.4.1 MediaPlayer

The Android multimedia framework includes support for playing variety of common media types, so that you can easily integrate audio, video and images into your applications. You can play audio or video from media files stored in your application's resources (raw resources), from standalone files in the file system, or from a data stream arriving over a network connection, all using MediaPlayer APIs. One of the most important components of the media framework is the MediaPlayer class. An object of this class can fetch, decode, and play both audio and video with minimal setup.

There are two ways to get a MediaPlayer object: one is to call MediaPlayer's static method create(), another one is to create by the key word new. The difference is the MediaPlayer object created by key word new is at state Idle while the object created by method create() is at state Prepared. The official documentation offers us a state diagram to understand the relationship between different states. The introductions of each state are as follows (see Figure 17-4):

- Idle: When a MediaPlayer object is just created using new or after reset() is called, it is in the Idle state. There is a subtle but important difference between a newly constructed MediaPlayer object and the MediaPlayer object after reset() is called. It is a programming error to invoke methods such as getCurrentPosition(), getDuration(), getVideoHeight(), getVideoWidth(), setAudioStreamType(int), setLooping(boolean),

setVolume(float, float), pause(), start(), stop(), seekTo(int), prepare() or prepareAsync() in the Idle state for both cases. If any of these methods is called right after a MediaPlayer object is constructed, the user supplied callback method OnErrorListener.onError() won't be called by the internal player engine and the object state remains unchanged; but if these methods are called right after reset(), the user supplied callback method OnErrorListener.onError() will be invoked by the internal player engine and the object will be transfered to the Error state.

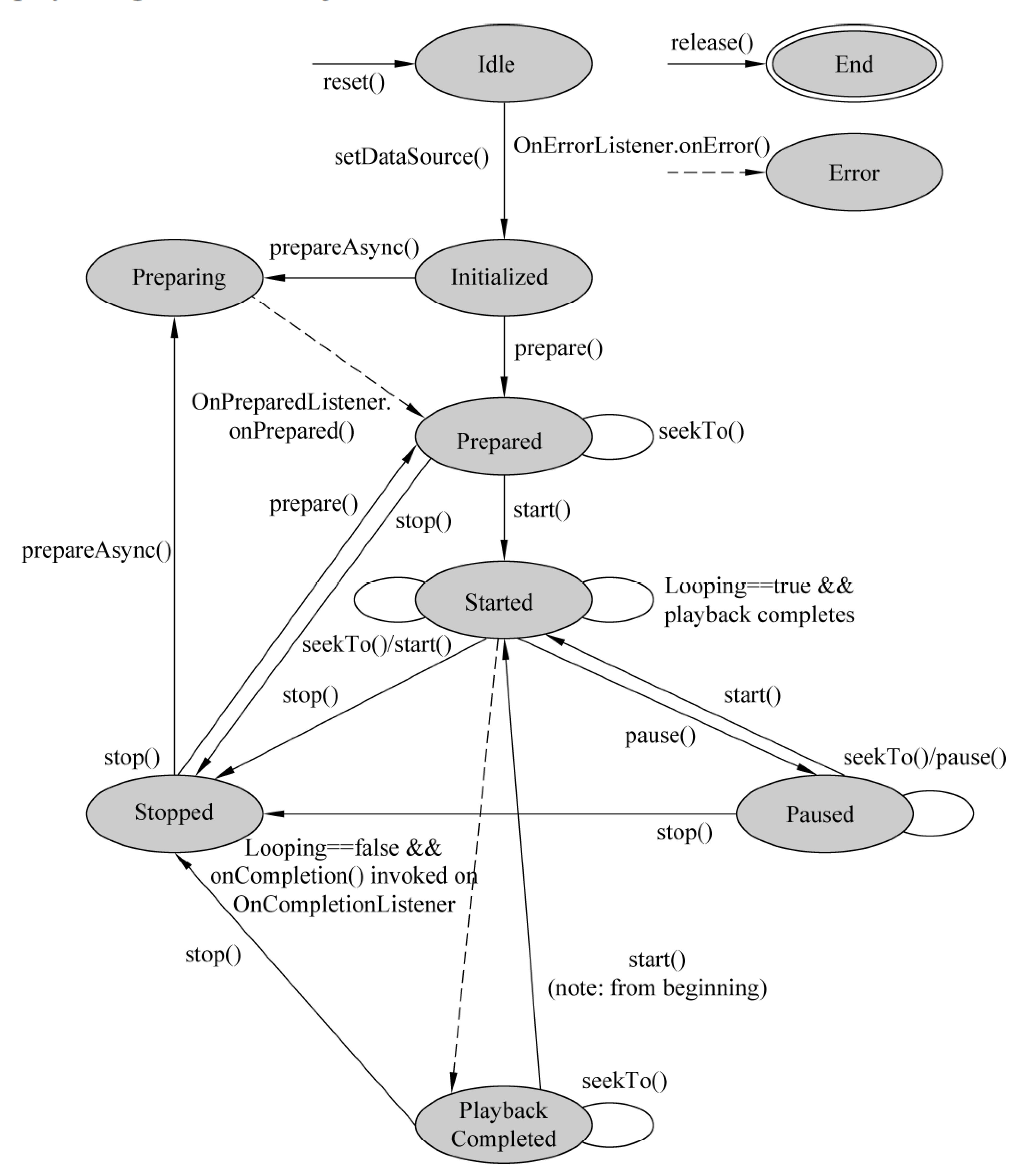


Figure 17-4 the states diagram of mediaplayer

- **End**: After release() is called, it is in the End state. Once a MediaPlayer object is no longer being used, call release() immediately so that resources used by the internal player engine associated with the MediaPlayer object can be released immediately. Once the MediaPlayer object is in the End state, it can no longer be used and there is no way to bring it back to any other state.

- **Initialized**: Calling setDataSource(FileDescriptor), or setDataSource(String), or setDataSource(Context, Uri), or setDataSource(FileDescriptor, long, long) transfers a MediaPlayer object in the Idle state to the Initialized state.
- **Prepared**: There are two ways (synchronous vs. asynchronous) that the Prepared state can be reached: either a call to prepare() (synchronous) which transfers the object to the Prepared state once the method call returns, or a call to prepareAsync() (asynchronous) which first transfers the object to the Preparing state after the call returns (which occurs almost right way) while the internal player engine continues working on the rest of preparation work until the preparation work completes. While in the Prepared state, properties such as audio/sound volume, screenOnWhilePlaying, looping can be adjusted by invoking the corresponding set methods.
- **Preparing**: The Preparing state is a transient state, and the behavior of calling any method with side effect while a MediaPlayer object is in the Preparing state is undefined.
- **Started**: To start the playback, start() must be called. After start() returns successfully, the MediaPlayer object is in the Started state. isPlaying() can be called to test whether the MediaPlayer object is in the Started state. When the playback reaches the end of stream, the playback completes. If the looping mode was being set to truewith setLooping(boolean), the MediaPlayer object shall remain in the Started state. While in the Started state, calling start() or seekTo() has not effect on a MediaPlayer object that is already in the Started state.
- **Paused**: Playback can be paused via pause(). When the call to pause() returns, the MediaPlayer object enters the Paused state. Calling start() to resume playback for a paused MediaPlayer object, and the resumed playback position is the same as where it was paused. When the call to start() returns, the paused MediaPlayer object goes back to the Started state. The method seekTo() can be called in this state, but the state will not change.
- **Stop**: Calling stop() stops playback and causes a MediaPlayer in the Started, Paused, Prepared or PlaybackCompleted state to enter the Stopped state. Once in the Stopped state, playback cannot be started until prepare() or prepareAsync() are called to set the MediaPlayer object to the Prepared state again.
- **PlaybackCompleted**: When the playback reaches the end of stream, the playback completes. If the looping mode was set to false , the player engine calls a user supplied callback method, OnCompletion.onCompletion(), if a OnCompletionListener is registered beforehand via setOnCompletionListener(OnCompletionListener). The invoke of the callback signals that the object is now in the PlaybackCompleted state. While in the PlaybackCompleted state, calling start() can restart the playback from the beginning of the audio/video source.

- **Error**: Once an error occurs, the MediaPlayer object enters the Error state, even if an error listener has not been registered by the application. Some playback control operation may fail due to various reasons, such as unsupported audio/video format, poorly interleaved audio/video, resolution too high, streaming timeout, and the like. Thus, error reporting and recovery is an important concern under these circumstances. Sometimes, due to programming errors, invoking a playback control operation in an invalid state may also occur. Under all these error conditions, the internal player engine invokes a user supplied OnErrorListener.onError() method if an OnErrorListener has been registered beforehand via setOnErrorListener (android.media.MediaPlayer.OnErrorListener). In order to reuse a MediaPlayer object that is in the Error state and recover from the error, reset() can be called to restore the object to its Idle state.

In order to monitor some events that may cause a state change, MediaPlayer offers us several methods to bind event listener, the most commonly used are the following four:

- public void setOnCompletionListener(MediaPlayer.OnCompletionListener listener): Register a callback to be invoked when the end of a media source has been reached during playback.
- public void setOnErrorListener (MediaPlayer.OnErrorListener listener): Register a callback to be invoked when an error has happened during an asynchronous operation.
- public void setOnPreparedListener (MediaPlayer.OnPreparedListener listener): Register a callback to be invoked when the media source is ready for playback.
- public void setOnSeekCompleteListener(MediaPlayer.OnSeekCompleteListener listener): Register a callback to be invoked when a seek operation has been completed.

This case mainly use MediaPlayer to play audio, for different audio sources, the play steps may be different. As follows:

(1) Play audio that's available as a local raw resource (saved in your application's res/raw/ directory).

a. Call method create(Context context,int resId) to load the specificed resource file.

b. Call method start(), pause(), stop() to play, pause and stop the music playback.

(2) Play original resource file(saved in your application's assets directory).

a. Call getAssets() of Context to get the AssetManager of this application.

b. Call method openFd(String name) of AssetManager object to open the specified original soundtrack resource, this method returns a AssetFileDescriptor object.

c. Call AssetManager's methods getFileDescriptor(), getStartOffset() and getLength() to get the audio file's FileDescriptor, starting position, length, etc.

d. Create a MediaPlayer object, call method setDataSource(FileDescriptor fd，long offset, long length) to load audio resources.

e. Call method prepare() to prepare audio.

f. Call method start(), pause(), stop() to play, pause and stop the music playback.

(3) Play audio file saved in the internal and external memory of phone which means the audio files are stored in the phone without any relations to application.

a. Create a MediaPlayer object, call method setDataSource(FileDescriptor fd, long offset, long length) to load audio resources.

b. Call method prepare() to prepare audio.

c. Call method start(), pause(), stop() to play, pause and stop the music playback.

(4) Playing from a remote URL via HTTP streaming.

a. Create a Uri object based on the position of the audio files on the network.

b. Create a MediaPlayer object, call method setDataSource(FileDescriptor fd, long offset, long length) to load corresponding audio resources according to the uri object.

c. Call method prepare() to prepare audio.

d. Call method start(), pause(), stop() to play, pause and stop the music playback.

The music in this case comes from phone's memory card, so we choose the third way, play according to the position of the audio playback.

17.4.2 Notifications

A notification is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Notification is sent via NotificationManager service. Similiar to dialog, create a Notification also need its internal class Builder. The class offers us methods like:

- **setAutoCancel(boolean autoCancel)**: Setting this flag will make it so the notification is automatically canceled when the user clicks it in the panel.
- **setDefaults(int defaults)**: Set the default notification options that will be used.
- **setContent(RemoteViews views)**: Supply a custom RemoteViews to use instead of the standard one.
- **setContentIntent(PendingIntent intent)**: Supply a PendingIntent to send when the notification is clicked.
- **setContentText(CharSequence text)**: Set the text (second row) of the notification, in a standard notification.
- **setContentTitle(CharSequence title)**: Set the title (first row) of the notification, in a standard notification.
- **setLargeIcon(Bitmap icon)**: Set the large icon that is shown in the ticker and notification.
- **setLights(int argb, int onMs, int offMs)**: Set the argb value that you would like the LED on the device to blink, as well as the rate.
- **setSound(Uri sound)**: Set the sound to play.

- **build()**: Combine all of the options that have been set and return a new Notification object.

The main components of Notification are shown in Figure 17-5:



Figure 17-5 structure diagram of notification

Main steps of sending a notification:

(1) Calling getSystemService() method to get the system NotificationManager services;

(2) Create a Notification object by Builder constructor;

(3) Set various properties for the Notification;

(4) Send Notification via NotificationManager.

Detailed code in this case to send notifications see MusicService line 98-111.

17.5 Thinking and Exercises

(1) The music can only be set as ringtone in this case, add functions that set music as notify ringtone and alarm sound.

(2) Which attribute is indispensable when configuring <provider> label in the manifest file? ( )

A. android:name B. android:authorities

C. android:exported D. A and B

(3) Which option is used to operate data exposed by ContentProvider? ( )

A. ContentValues B. ContentResolver

C. URI D. Context

(4) Which object calls method query to read data when operate data shared by ContenProvider? ( )

A. ContentResolver B. ContentProvider

C. SQLiteDatabase D. SQLiteHelper

(5) Which callback method is not belong to Service lifecycle? ( )

A. onCreate() B. onBind() C. onStart() D. onStop()

(6) Which method must be implemented when developing a custom Service component? ( )

A. onCreate() B. onBind() C. onStartCommand() D. onUnbind()

(7) Which statement about start Service through startService() and bindService() is false? ( )

A. once the service started by startService(), there is no relations between visitor and Service, while the service started by bindService() will pervish with the visitor

B. the Service started by startService() will call method onStartCommand() automatically while the Service started by bindService() will call onBind() automatically

C. the Service started by startService() cannot communicate and transfer data with vistors while the Service started by bindService() can

D. the Service started by bindService() must implements method onBind() while the Service started by startService() do not have this requirement

(8) Which description about BroadcastReceiver is correct? ( )

A. BroadcastReceiver can only be registered in manifest file

B. BroadcastReceiver can not be destroyed after registration

C. BroadcastReceiver can only receive custom broadcast message

D. BroadcastReceiver can be registered and destroyed in Activity independently

(9) Which method is needed to complete the preparatory work before play audio and video resources by MediaPlayer? ( )

A. setDataSource() B. prepare() C. begin() D. ready()

(10) The steps of playing music file in the external memory card by MediaPlayer is ( ).

A. call method MediaPlayer.create() and pass the file path to the method, then get the MediaPlayer object. Finally, get ready to play

B. pass the path of music file to the MediaPlayer's constructor method, then get ready to play

C. create a MediaPlayer object, call method setDataSource to set data source, then get ready to play

D. both A and C

Appendix A The Common Errors and Debugging Methods

Android is based on the Java language. Some simple grammar errors will be prompted automatically by IDE and developers can correct it quickly according to the prompted information. Compiled successfully doesn't means that the program can run normally. A runtime exception may occur at run time and lead to exit. There is a mistake called logic errors. The program can run normally when this error occurs, but the result is inconsistent and unexpected. Here we will make a brief introduction aims at the solution of the two scenarios later.

1. Programming debugging methods

(1) Logcat output log information.

On the Android platform, we can use the Log class, add some "records" in the program code and view the record through the "LogCat" Eclipse tools. When the program executes the "record", a corresponding "record" will output a message in LogCat. Developers can check the program execution process's consistence with our expectations by analyzing these records. You can judge the potential for error in the program code area so that the accurate position.

Open the LogCat view:

The Eclipse menu select **Windows→Show View→Other→Android→LogCat,** LogCat window appears in the console window. As shown in the figure below in Figure A-1.

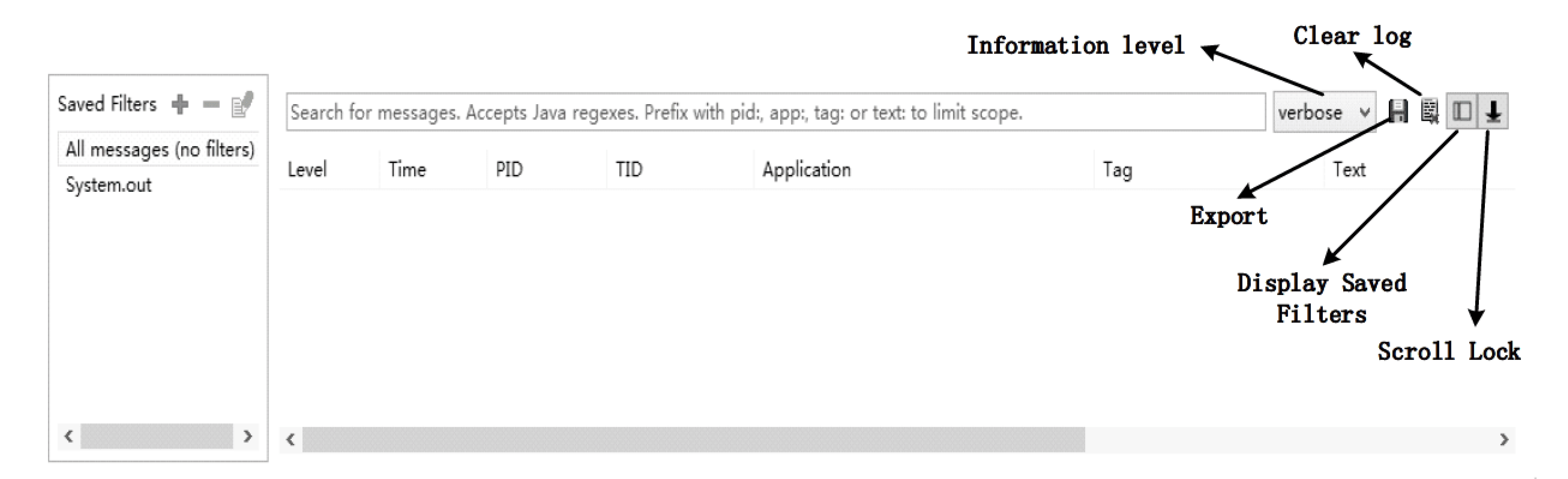
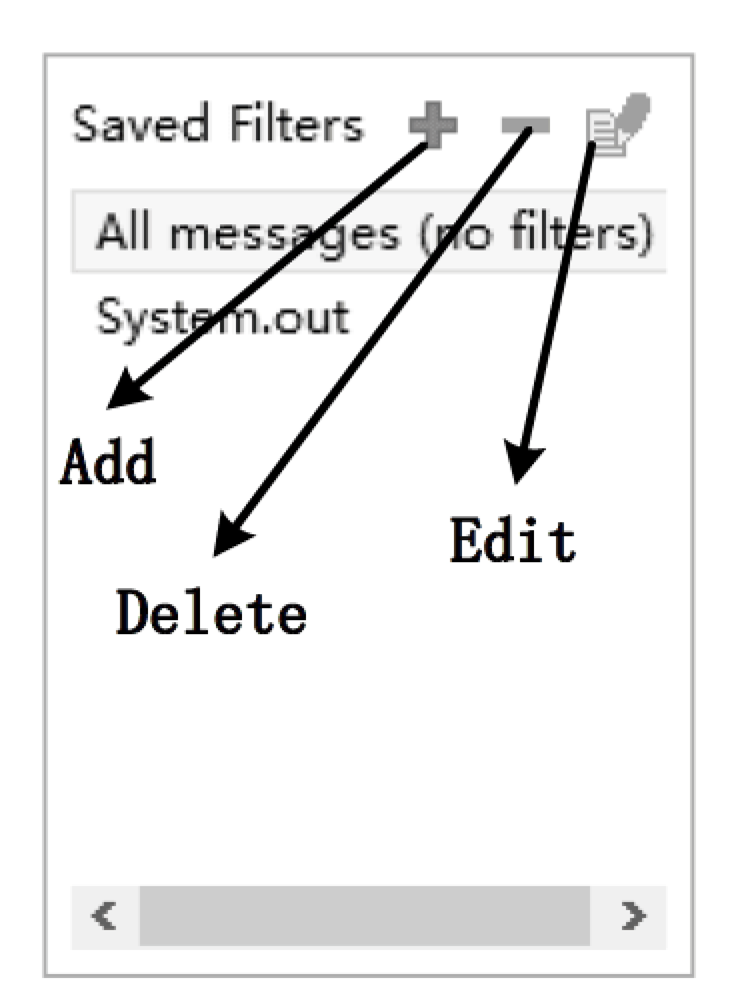


Figure A-1 the figure of the analysis of LogCat user interface

By default, there are many information shown in the LogCat . In order to show the

messages you need, you can filter the information like Figure A-2.



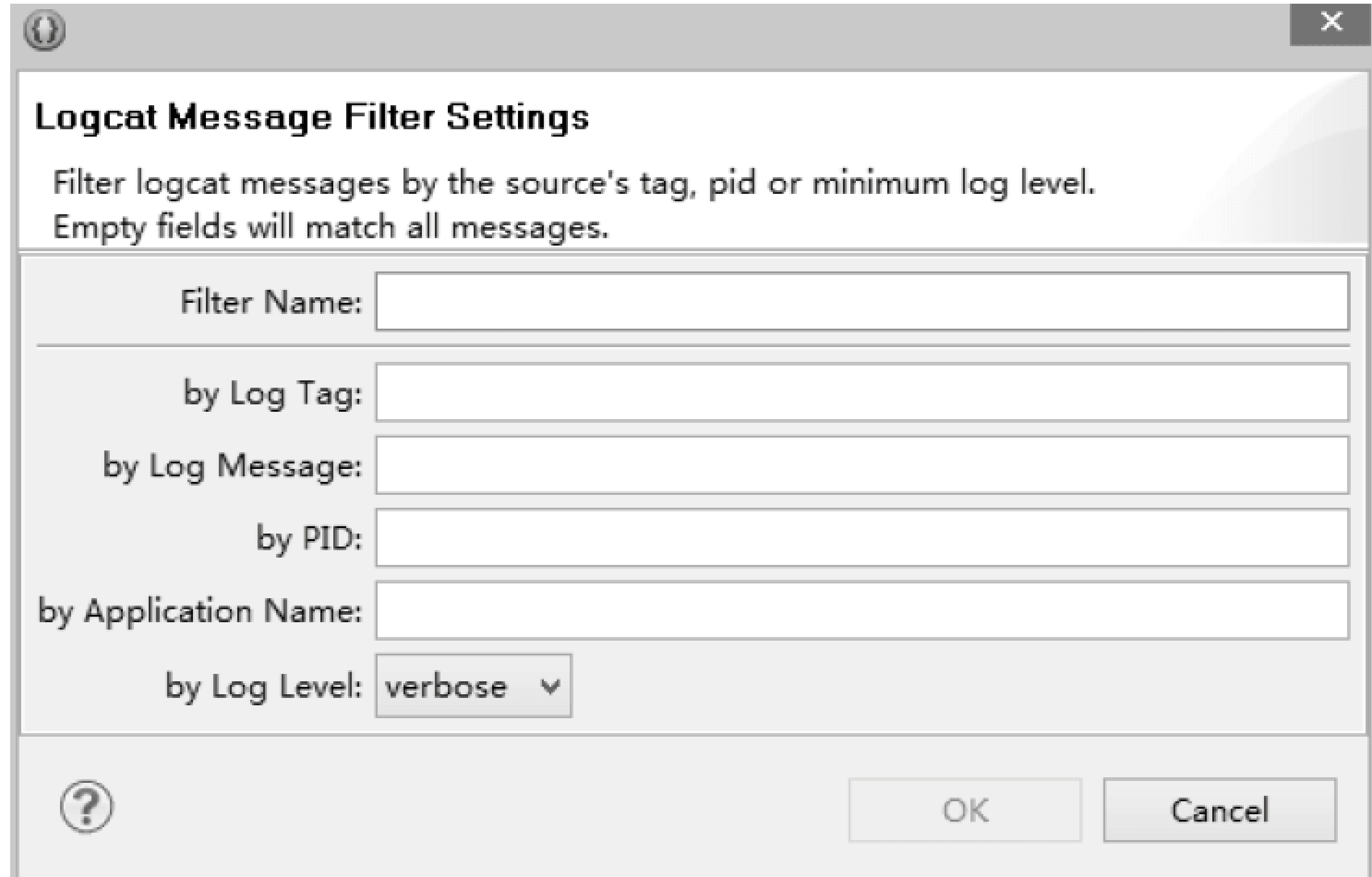


Figure A-2 the figure of the analysis of filter in LogCat

android. util. log common methods include: the v (), the d (), Log. i (), the w() and the e(). According to the first letter corresponding to the VERBOSE, DEBUG, INFO, WARN, ERROR.

Table A-1 shows all methods to display message and corresponding display color.

The order in terms of verbosity, from least to most is **ERROR, WARN, INFO, DEBUG, VERBOSE**. Verbose should never be compiled into an application except during development. The DEBUG logs are compiled in but stripped at runtime. ERROR, WARN and INFO logs are always kept.

Table A-1 message level and corresponding display color

methods	color	message level
Log.v()	Black	Any message(VERBOSE)
Log.d()	Blue	Debugging message(DEBUG)
Log.i()	Green	Reminder message(INFO)
Log.w()	Orange	Warning message(WARN)
Log.e()	Red	Errors message (ERROR)

The related methods in Log class usually need two parameters, one is the Tag information, named the Tag, and another is the content of the information. We can use the Tag to filter, rapidly posit to the log information.

Notice: sometimes the LogCat will not show any information.

Solution: in DDMS→devices view, select the operation of the equipment, or to open the LogCat, or restart the Eclipse.

A simple example: (console print log information sequence).

Define two classes: Person.java and Student. java.
Person.java

```
import android.util.Log;
```

```
public class Person {
    public Person(){
        Log.i(MainActivity.TAG, "Person Construtor invoked!");
    }
    public void say(){
        Log.i(MainActivity.TAG,"Person say() invoked!");
        System.out.println("I'm a super class!");
    }
}
```

Student.java

```
public class Student extends Person {
    private String name;
    public Student(){
        this("unknown");
        Log.i(MainActivity.TAG, "Student Constructor without argument invoked!");
    }
    public Student(String name){
        this.name=name;
        Log.i(MainActivity.TAG, "Student Constructor with a argument invoked!");
    }
    public void say(){
        Log.i(MainActivity.TAG,"Student say() invoked!");
        System.out.println("I'm a subclass of Person! My name is "+name);
    }
}
```

MainActivity.java

```
public class MainActivity extends Activity {
    public static final String TAG="LogCatInfoTest";
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Person person=new Student();
        person.say();
    }
}
```

What is the output sequence of the console log information? (Select output information, and sort them)

① Person Constructor invoked!

② Person say() invoked!

③ I'm a super class!

④ Student Constructor without argument invoked!

⑤ Student Constructor with a argument invoked!

⑥ Student say () invoked!

⑦ I'm a subclass of Person! My name is Xxx

Consequence: ①→⑤→④→⑥→⑦

(2) The Debug function provides by Eclipse.

First of all, you can set breakpoints in the code. When the program meets the breakpoint, it will stop. The methods of setting a breakpoint method are:

① Double click on the left in the code line number, generating breakpoint symbol.

② The mouse on line code lives in, right-click it and select the first Toggle Breakpoint, generating Breakpoint symbol.

③ Place the cursor in line need to add breakpoints, and then press Ctrl + Shift + B, the breakpoint symbol can generate.

If you want to cancel the corresponding breakpoint, simply repeat the above operation.

Run the program after setting the breakpoint. At this time it is no longer to select Run As rather than Debug As. The program will stop when performing the breakpoint, and jump to the Debug view in Figure A-3.

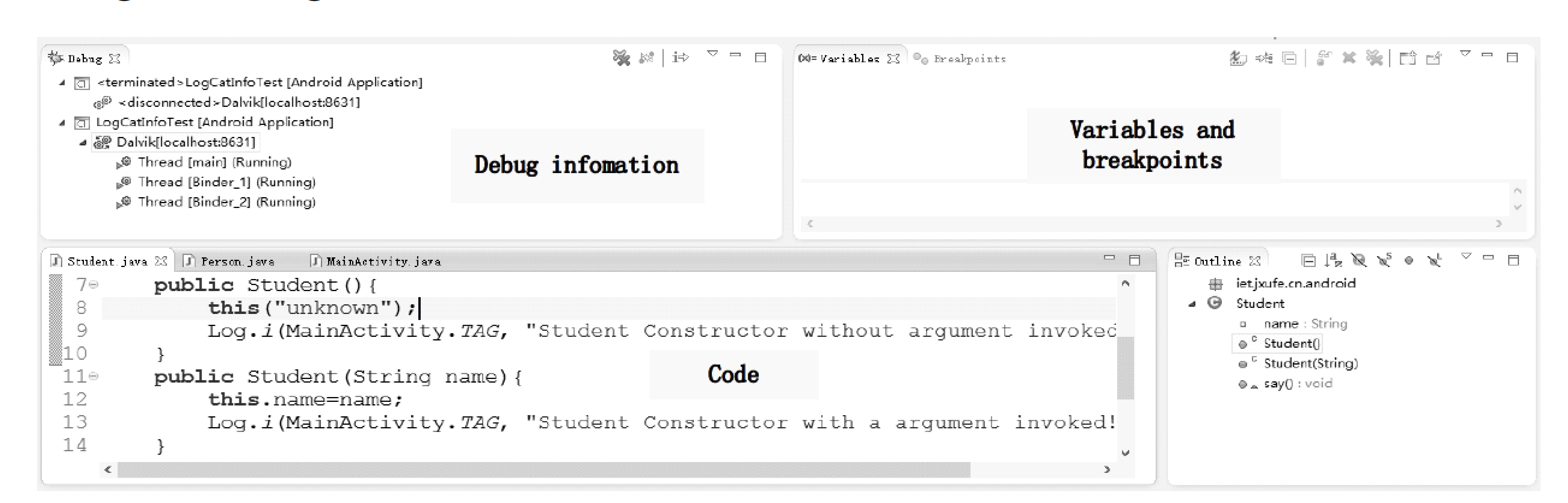


Figure A-3 the figure of views of debugging

Some shortcut keys for debug:

Start the Debug:	F11
Step into (perform inside)	F5
Step over (execute next)	F6
Step Return (return)	F7
Perform the end	F8

2. Common mistakes when running

(1) The null pointer exception (called an object of the value is null method or member variables)

A. a reference type variable only statement, definition, no initialization, the default

value is null.

```
public class MainActivity extends Activity {
    private Button login;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        login.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                System.out.println("The Login Button had been clicked!");
            }
        });
    }
}
```

At this point, the compiler doesn't have any mistake, but the program will throw null pointer exception! Because it hasn't a specific value from login model and its defaulted value is null in Figure A-4.

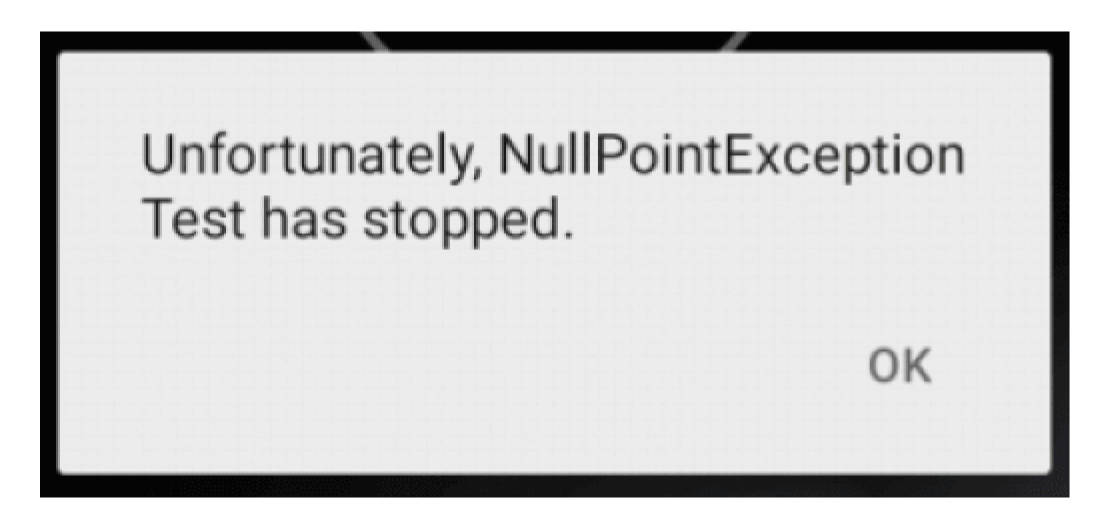


Figure A-4 the figure of throwing NullPointException

You need check description of the error information in the console, generally speaking, the beginning of the first to see the error, the description of the error, such as Figure A-5.

```
FATAL EXCEPTION: main
java.lang.RuntimeException: Unable to start activit
y ComponentInfo{iet.jxufe.cn.android/iet.jxufe.cn.a
ndroid.MainActivity}: java.lang.NullPointerExceptio
n
```

Figure A-5 the figure of detail error information in LogCat

Then check Caused by statements and find the reason, such as Figure A-6.

```
Caused by: java.lang.NullPointerException
at iet.jxufe.cn.android.MainActivity.onCreate(Main
Activity.java:14)
```

Figure A-6 the figure of reason information

Find reasons and then need to analyze it, why is null values, and thus for the corresponding modification.

Is the modified line? Why is that?

```
public class MainActivity extends Activity {
    private Button login=(Button)findViewById(R.id.login);
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //login=(Button)findViewById(R.id.login);
        login.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                System.out.println("The Login Button had been clicked!");
            }
        });
    }
}
```

At this point, the system still throws null pointer exception, it is because the findViewById() method use Id to find the appropriate widget from a layout file, it is the premise of the layout file has been loaded. And layout file is loaded in the onCreate() method, and the login as a member variable, is at the time of class loading is performed, and the onCreate() method is after creating the object of the class. The correct approach:

```
public class MainActivity extends Activity {
    private Button login;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        login=(Button)findViewById(R.id.login);
        login.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                System.out.println("The Login Button had been clicked!");
            }
        });
    }
}
```

B. according to the findViewById () method is unable to find corresponding widget (mainly for multiple layout file)

```
public class MainActivity extends Activity {
    private Button login;
    private Button reset;
    private EditText name,psd;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        login=(Button)findViewById(R.id.login);
        login.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Builder builder=new AlertDialog.Builder(MainActivity.this);
                builder.setTitle("Welcome to login");
                View view=getLayoutInflater().inflate(R.layout.login,null);
                reset=(Button)findViewById(R.id.reset);
                name=(EditText)findViewById(R.id.name);
                psd=(EditText)findViewById(R.id.psd);
                reset.setOnClickListener(new OnClickListener() {
                    public void onClick(View v) {
                        name.setText("");
                        psd.setText("");
                    }
                });
                builder.setView(view);
                builder.create().show();
            }
        });
    }
}
```

The above italic part is changed to:

reset=(Button)view.findViewById(R.id.reset);

name=(EditText)view.findViewById(R.id.name);

psd=(EditText)view.findViewById(R.id.psd);

(2) Type conversion exceptions(caused by types compatible)

The findViewById () method returns the value of the View object, and we need invoke some special method of the components, we must turn the View objects to a specific subclass object, by the superclass casts for the subclass object, is not going to go wrong at compile time, but when running, if the specific objects do not agree with what you convert object type, there is no parent-child relationships, will appear a runtime exception. For example, Converting ImageView forced into TextView, converting TextView into Button, and Button cast TextView will not go wrong, because TextView is the parent class of Button, subclass object reference can be assigned to the superclass.

(3) Array bounds exceptions

In order to avoid the array bound during the process of implement drop fission and cycle show picture, we usually use length attribute of array to judge, as well as to the length of the array modulus, so can only take 0 to length - 1 value, not crossing the line.

(4) Repeated warnings to run the program

The current program has been running in the foreground, and the program does not have any updates, will now run repeatedly hint warning:

Warning: Activity not started, its current task has had been brought to the front.

Solution: a, exit the program to run; B, modify the program to run again, such as adding a space.

(5) Android runtime exception android.view.InflateException: Binary XML file line #: Error inflating class Xxxx.

The reason for the errors:

① quote the name of the class problem is the name of the tag to write wrong, this system according to the reflection mechanism can't find the corresponding class;

② if it is a custom tag, so the custom attributes of the class must implement contains constructor;

View(Context context) //Simple constructor to use when creating a view from code

View(Context context, AttributeSet attrs)

//Constructor that is called when inflating a view from XML

```
public class MyButton extends Button {
    public MyButton(Context context) {
        super(context);
    }
//  public MyButton(Context context, AttributeSet attrs) {
//      super(context, attrs);
//  }
}
```

(6) The method of setContentView() failed when using ListActivity

When using a ListActivity, the activity can not contain any layout file, or call the setContentView () method, if you are using the setContentView () method to set the display interface, the layout file must contain a ListView, and ListView with id: @ android: id/list. Otherwise a runtime Exception will be thrown (Fatal Exception): Your content must hava a ListView whose id attribute is' android.R.id.list '. Why is that?

ListActivity has a default layout that consists of a single, full-screen list in the center of the screen. However, if you desire, you can customize the screen layout by setting your own view layout with setContentView() in onCreate(). To do this, your own view must contain a ListView object with the id "@android:id/list".

(7) When importing the project into Eclipse, almost all the Java classes are error.

This phenomenon is usually caused by the version of Android.originally used by the project version does not exist in the unit, at this point we can see in the project document structure does not exist in the Android development kit.

Solution: for the project introduced the Android development kit, not necessarily, and the original versions, can introduce development kit is higher than the original version. Operation process: the project selected right→select properties pop-up dialog → choose Android, then on the right to choose an existing Android development kit →Apply →OK.

(8) When import projects into Eclipse it often shows: Android requires compiler the compliance level 5.0 or 6.0, Found '1.4' home. Both Please use the Android Tools > Fix the Project Properties.

Solution:

① right-click on the engineering document according to directions →Android Tools →Fix Project Properties;

② if ① is invalid, then manually open Project→Properties → javaCompiler → selected Enable Project specific setting → again choose the Compiler Compliance Level (choose any one of the default value) → OK;

③ repeat the step 2, the Compiler around Level selected to the correct value → the value is generally currently installed JDK version, such as the JDK 5 corresponds to 1.5, the JDK 6 corresponding to 1.6), OK.

(9) import android project @ Override error

Problem description: when sometimes import android project, it is just used no problem project, but got an error to import.

Tips: The method... Must override a spuerclass method, then the eclipse gives us hints let us remove @ override.

The error from the Java compiler is no @ Override Java1.5, 1.6.

Solution: make eclipse use java1.6 instead of 1.5.

Operation process is as follows:

Choose Window of Eclipse→Preferences→Java Compiler

Although we could see in the right of the Compiler compliance level chosen is 1.6 right now, but it may not worked for every project, so we continue to choose "Configure Project Specific Settings...", then we can see our engineering, and choose an error engineering→OK

Here the JDK compliance is not 1.6, you should change it to 1.6 →OK.

Appendix B The Knowledge of Android Programming

1. Android environment building and program structure analysis

(1) The command in Android to start simulator (Android Virtual Device) is ( ).

A. adb B. android C. avd D. emulator

(2) The command in Android which in charge of copy and install application between simulator file and computer file is ( ).

A. adb. B. android C. avd D. emulator

(3) The simulator creating command is ( ).

A. the android create avd n (the name of the simulator) t (android version)

B. the adb create avd n (the name of the simulator) t (android version)

C. avd create avd n (the name of the simulator) t (android version)

D. emulator creates avd n (the name of the simulator) t (android version)

(4) Which description about the assets catalog and res directory is not correct? ( )

A. assets directory can be arbitrary set up a folder, in the preservation of resources will be intact in the installation package, will not be compiled into the binary

B. the res directory of resources judge whether to be used in packaging, unused resources will not be packaged into the installation package

C. assets directory and the resources of the res directory in R.java generated in the resource tag

D. the res directory includes only some fixed sub folders, unable to create a folder

(5) Which description of res/raw directory is correct? ( )

A. the files in the directory will be intact storage on the device will not converted to binary format

B. the files in the directory will be intact storage would be converted to binary format to the device

C. the files in the directory with or without using stored in the installation package is intact

D. the files in the directory not R.java generated in the resource tag

(6) The extension name of AndroidManifest file is ( ).

A. jar　　B. XML　　C. apk　　D. Java

(7) Which statement of the AndroidManifest files is correct? (　)

A. AndroidManifest listing file is necessary for each Android project, it is the global description file of the Android applications

B. AndroidManifest listing file illustrates the applications, the use of the name of the icon, and contains components, etc.

C. AndroidManifest manifest file contains the application uses system permissions required statement, also contains other program to access the required permissions for the statement

D. AndroidManifest listing file of the root element is < application >, the contained components such as Activity, Service, and so on are included in the < application > element

(8) Which description of the content of AndroidManifest file is correct? (　)

A. declare the authority should be on the application itself need < application > element

B. statement invokes the application the required permission should be put in the < application > element

C. through the function keys, to view a mobile phone application software, the function in listing application of labels can pass the < application > element's android: label property set

D. through function keys, can see on the mobile phone application software, mobile phone function in listing application tag can be through the main Activity of android: label property set

(9) The four components of the Android usually need register in AndroidManifest file before using, which of the following components can be used without registration manifest file? (　)

A. Activity　　B. Service

C. ContentProvider　　D. BroadcastReceiver

(10) Which description is not correct? (　)

A. gen Android applications directory R.java will be deleted after automatically generated

B. Android project res directory is a special directory, to hold the application of various resources, naming rules can support lowercase letters (a-z, a-z), Numbers (0-9) and horizontal (_)

C. AndroidManifest listing file is necessity for each Android projects, is to use global description of the project, through the package name + component can specify the full path of the component

D. Android project assets and res can deposit resource file directory, but unlike res

assets support arbitrary depth of subdirectories, R.java file in it won't be able to generate any resource ID in the Java

(11) Which of the following does not belong to the Android application layer in the architecture? ( )

A. address book. B. calendar C. SQLite D. SMS program

(12) Which configuration is not correct when register components in the manifest file? ( )

A. < activity android: name = ". MyActivity ">
< intent - filter >
< action android: name = "iet.jxufe.cn.action.MyActivity" / >
< / intent - filter >
< / activity >

B. <service android: name = "MyService" > < / service >

C. <provider android: name = "MyProvider" > < / provider >

D. < receiver android: name = "MyReceiver" >
< intent - filter >
< action android: name = "iet. jxufe.cn. receiver. MyReceiver" / >
< / intent - filter >
< / receiver >

2. Android interface programming

(1) Which widget is not directly or indirectly inherited from the ViewGroup class? ( )

A. GridView B. ListView C. ImageView D. ImageSwitcher

(2) Which option does not belong to the Android layout manager? ( )

A. FrameLayout B. GridLayout C. BorderLayout D. TableLayout

(3) The unit of setting the text size in Android is recommended to use ( ).

A. px B. dp C. sp D. pt

(4) Which statement about TextView and ImageView is true? ( )

A. TextView is mainly used to display text, can set the text size, color and so on TextView in addition to setting the background image, unable to display images on it

B. ImageView is mainly used to display images, can set the source of the image, the zoom type, etc., can't display text on your ImageView

C. ImageView from TextView inherited, it is the extension of TextView

D. in the ImageView tag set android: text property, can complain directly

(5) Which option does not have inheritance? ( )

A. TextView, AutoCompleteTextView B. TextView, Button

C. ImageView, ImageSwitcher D. ImageView, ImageButton

(6) In a horizontal linear layout, which property can make the width of control widget

shows a certain proportion? (　)

A. android: layout_width　　B. and android: layout_weight

C. android: layout_margin　　D. android: layout_gravity

(7) The following attributes, do not belong to the EditText is (　).

A. android: inputType　　B. android: hint

C. android: scaleType　　D. android: minLines

(8) Which description of the LinearLayout is correct? (　)

A. LinearLayout, level of all the controls are according to the horizontal direction in A next to A permutation.beyond the width of the screen, will automatically generate horizontal scroll bar, drag the scroll bar to view the other controls

B. horizontal LinearLayout of all the controls is according to the horizontal direction in a next to a permutation, beyond the width of the screen, will automatically display a new line of other controls

C. horizontal LinearLayout of all the controls is according to the horizontal direction in a next to a permutation, beyond the width of the screen, will not be displayed redundant control

D. horizontal LinearLayout of all the controls are according to the horizontal direction in a next to a permutation, beyond the width of the screen, add controls, the program is run times wrong

(9) Which description about TableLayout is not correct? (　)

A. TableLayout is inheritance from linear layout

B. can be clearly specified in TableLayout contains many column lines

C. In TableLayout a widget can account for multiple columns

D. if the control is added directly to the TableLayout, not added in the TableRow, then the control will separate a line

(10) The right way to set a listed as a retractable column in the table layout is (　).

A. set the TableLayout properties: android: stretchColumns = "x", the serial number of x column

B. set the TableLayout properties: android: shrinkColumns = "x", the serial number of x column

C. setting specific properties: android: stretchable = "true"

D. set up the concrete column attributes: android: shrinkable = "true"

(11) In the RelativeLayout, if we want to make a control widget center, we can use (　).

A. android: gravity = "center"

B. android: layout_gravity = "center"

C. android: layout_centerInParent = "true"

D. android: scaleType = "center"

(12) Which attributes value can only be true or false in RelativeLayout? (　)

A. android: layout_alignTop B. android: layout_alignParentTop
C. android: layout_toLeftOf D. android: layout_above

(13) The method to set ImageButton's background transparent successfully is ( ).

A. set ImageButton android: alpha attribute value is 0
B. set ImageButton android: alpha attribute value is 255
C. set the ImageButton android: background of attribute value is: # FFFFFFFF
D. set ImageButton android: background of attribute value is: # 00000000

(14) If the image size is 1200 * 1200, are now displays it on a 300 * 200 ImageView, if set the ImageView scaleType attribute has a value of fitCenter, image scaling as ( ).

A. such as scaling, scaling for 4
B. such as scaling, scaling is 6
C. transverse zoom ratio as 6, the vertical axis scaling of 4
D. inverse scaling is 4, the vertical axis zoom ratio as 6

(15) Which method doesn't need to rewrite when customize an Adapter inherits from BaseAdapter? ( )

A. getCount() B. getView() C. the getItem() D. getDropDownView

(16) Android contains many Adapter related classes, which class is not inherited from BaseAdapter?( )

A. ArrayAdapter B. SimpleAdapter C. CursorAdapter D. PagerAdapter

(17) The following description about SimpleAdapter construction method in parameter is incorrect ( ).

A. the first parameter to the Context object, usually you just need to deliver current object of the Activity
B. the second parameter to the list of data sources, can be either an array, or it can be a set
C. the third parameter layout file for each item in the list, can contain multiple controls in this layout
D. there is a one-to-one relationship between the fourth and fifth parameter, according to the fourth parameter acquisition of data, will be in the fifth, according to control parameters are specified and the elements in the fifth parameters, must be in the third argument in the specified layout file

(18) AutoCompleteTextView automatically input controls, according to the content of user input, to match from the specified data source all qualified data, and displayed in the following drop-down list, which allows users to choose. The following property, which can set up the minimum number of characters required for user input? ( )

A. android: completionThrehold B. android: completionHint
C. android: dropDownVerticalOffset D. android: dropDownHorizontalOffset

(19) Which option represents a drag control widget? ( )

A. RatingBar, B. ProgressBar C. SeekBar D. ScrollBar

(20) Which option can not be set directly by the RatingBar attributes? ( )

A. five-pointed star number B. current fraction

C. score increment of D. the color of the pentagram

(21) The maximum number of child of a vertical ScrollView is? ( )

A. 0 B. 1 C. 2 D. countless

(22) Which control widget is not inherited from the Button? ( )

A. ImageButton B. RadioButton C. CheckBox D. ToggleButton

3. The Android dialog and menu

(1) Which description is not correct about AlertDialog? ( )

A. AlertDialog show () method can create and display A dialog

B. the create () and show () method of AlertDialog. Builder returns the AlertDialog object

C. AlertDialog can not directly use the new keyword build objects, and values must use its Builder

D. AlertDialog. Builder of the show () method can create and display the dialog

(2) Build AlertDialog need inner class Builder, the Builder class contains a lot of methods, the following methods, the method return type and other items ( ).

A. create () B. setMessage () C. setView () D. setAdapter ()

(3) How many Buttous can an AlertDialog dialog have? ( )

A. 1 B. 2 C. 3 D. countless

(4) To customize a dialog,the method of adding View object to the current dialog is ( ).

A. setDrawable B. setContent () C. setAdapter() D. setView ()

(5) If you need to create options menu in the Android, which method you must rewrite in the Activity? ( )

A. onCreateOptionsMenu () B. onCreateContextMenu ()

C. onOptionsCreateMenu () D. onContextCreateMenu ()

(6) In the menu resource file, which label is unable to identify? ( )

A. < menu > B. < item > C. < submenu > D. < group >

(7) The method of creating a submenu to the menu is ( ).

A. add B. addMenu C. addSubMenu D. addMenuItem

(8) Processing method of the following events, not suitable for the click event of processing options menu item is ( ).

A. using the method of onOptionsItemSelected (MenuItem item) to deal with

B. using the method of onContextItemSelected (MenuItem item) to deal with

C. using the method of OnMenuItemClickListener onMenuItemClick (MenuItem item) to deal with

D. using the method of onMenuItemSelected (int featureId, MenuItem item) to deal

with

4. The Android event handling

(1) Which kind of design patterns based on the monitoring of event handling mechanism is applied to Android event handling mechanism? ( )

A. the observer pattern B. proxy mode

C. strategy pattern D. decorator pattern

(2) The method to add a monitor for CheckBox whether the selected event listener is ( ).

A. setOnClickListener B. setOnCheckedChangeListener

C. setOnMenultemSelectedListener D. setOnCheckedListener

(3) When using asynchronous task processing time consuming operation, the Android system provides us with AsyncTask abstract class, which method we must implement?( )

A. onPreExecute B. the doInBackground ()

C. onPostExecute () D. onProgressUpdate ()

(4) Use an asynchronous task processing time.which method cannot change the user interface? ( )

A. onPreExecute () B. doInBackground ()

C. onPostExecute () D. onProgressUpdate ()

(5) Which statement about creating Message object is not correct ( ).

A. Message msg = new Message ();

B. Message msg = Message. Obtain ();

C. Message msg = Message. Obtain (Message message);

D. Message msg = Message. CopyFrom (Message message);

5. The Android resource definitions

(1) The following files in the Android project of the res/drawable folder.an error or not directly in R.java file generated in the member variables is ( ).

A. aaa. XML B. BBB. JPG C. ccc. JPG D. ddd. eee. JPG

(2) Which option will not make an error when putting into the res/drawable folder? ( )

A. my_picture.PNG B. myDog.JPG.

C. myCat PNG D. 9 _dog. JPG

(3) Which option of the color value is illegal?( )

A. #ggg B. #ffff C. #eeeeee D. #dddddddd

(4) Use the Android Canvas class drawRect (10,10,20,20, new Paint ()), draw a rectangle, the rectangle area is ( )

A. 100 B. 200 C. 300 D. 400

(5) The Android constants defined in some resources, is usually placed in the <resources> tag, which of the following does not belong to a < resource > tag child tags? ( )

A. <string> B. <color> C. <drawable> D. <object-array>

(6) The right way to customize style is ( ).

A. < resources >

< style name = "myStyle" >

< item name = "android: layout_width" > match_parent < item >

< / style >

< resources >

B. < style name = "myStyle" >

< item name = "android: layout_width" > match_parent < item >

< / style >

C. < resources >

< item name = "android: layout_width" > match_parent < item >

< resources >

D. < resources >

< style name = "android: layout_width" > match_parent < / style >

< resources >

(7) In android, ImageButton not only can be a JPG, PNG format image files, can also be a XML file defines the image, if you need to define a as the button state changes of the XML file pictures, the file is the root element (   ).

A. <animation-list>   B. <layer-list>

C. <selector>   D. <shape>

(8) Which type of Drawable object, can achieve the spread effect?(   )

A. StateListDrawable   B. LayerDrawable

C. ShapeDrawable   D. ClipDrawable

(9) Which option is the root element to define Tweens in the XML file? (   )

A. <set>   B. <animation-list>

C. <layer-list>   D. <selector>

(10) The correct description of the XML resource file below is (   ).

```
<? The XML version = "1.0" encoding = "utf-8"? >
< shape XMLNS: android = "http://schemas.android.com/apk/res/android"
        android: shape = "line" >
        < stroke
        android: color = "@ color/gray"
        android: dashWidth = "5 dp"
        android: dashGap = "3 dp" / >
< / shape >
```

A. this is to draw a shape file width of 5 dp, high color piece for 3 dp

B. this is to draw a shape file width from 5 dp to 3 dp isosceles trapezoid

C. the shape file is to draw a bottom for 5 dp high for 3 dp isosceles triangle

D. the shape file is to draw a dotted line and the solid line 5 dp, interval of 3 dp

6. One of Android four major components: the Activity

(1) Which option does not belong to the Activity of the launchMode attribute of the attribute value? ( )

A. singleStack B. singleTop C. singleTask D. singleInstance

(2) Which option is not the Activity start method? ( )

A. startActivity B. goToActivity

C. startActivityForResult D. startActivityFromFragment

(3) Suppose set MainActivity lauchMode attribute value to singleInstance.and MainActivity already exists in the stack, the current Activity at this time to jump to MainActivity, what method will be the first to call MainActivity? ( )

A. onCreate() B. onResume() C. onNewIntent() D.onSaveInstanceState()

(4) Which method does not exist in the life cycle of Activity? ( )

A. onCreate () B. onStart () C. onStop () D. onFinish ()

(5) Which option is indispensable when configuring the Activity? ( )

A. android:name B. <action…/>

C. <intent-filter…/> D. <category…/

(6) About the entry Activity the application, among the following description, which one is incorrect? ( )

A. each application has one and only one entrance to the Activity.no entrance to the application of the Activity; the runtime will be an error

B. entrance Activity < intent - filter.../> You can have multiple <action… / > element tag

C. entrance Activity < intent - filter.../> You can have multiple < category > element tag

D. entrance Activity < intent - filter.../ > element must have a < action android: name = "Android. Intent. Action. MAIN" / > element, and has a < category android: name = "Android. Intent. The category. The LAUNCHER" / > element

(7) In the manifest file, configure the Activity, which tags can't in < intent - filter... / > tag identification? ( )

A. <action…> B. <category…/>

C. <data…/> D. <type…>

(8) Which statement about < intent - filter... / > tag is not correct ?( )

A. the tag can contain 0 - N < action... / > tag

B. the tag can contain 0 - N < category... / > tag

C. the tag can contain 0 - N < data... / > tag

D. the system will accord the label to determine when to start the component

7. Data store in the Android

(1) The first called method when reading the phone storage space is ( ).

A. openFileOutput()　　B. read()

C. write()　　D. openFileInput()

(2) SharedPreferences file path and extensions is (　).

A. / data/data/shared_prefs / *. TXT

B. / data/data/package name/shared_prefs / *. XML

C. / MNT/sdcard/specify folders specified extensions

D. any path/any extension

(3) For an existing userPreference SharedPreferences object, want to deposited in the "name", a string of userPreference should call what method (　).

A. edit ()　　B. save ()　　C. commit ()　　D. putString ()

(4) Which data type is not supported by SQLite? (　)

A. BLOB　　B. INTEGER　　C. VARCHAR　　D. REAL

(5) Among the following description about SQLiteOpenHelper, which one is incorrect? (　)

A. SQLiteOpenHelper available in Android database management tools, it is mainly used for database creation, open, version updates, etc., it is an abstract class

B. inheritance SQLiteOpenHelper class, must override the onCreate () method

C. inheritance SQLiteOpenHelper class, must override the onUpgrade () method

D. inheritance SQLiteOpenHelper class that can provide a constructor can also do not provide a constructor

(6) SQLiteOpenHelper available in Android database management tools, used to manage database creation, version updates, open, etc., it is an abstract class, if you create a subclass of the class, the following method, which is not must be included in the newly created class? (　)

A. method　　B. onCreate ()　　C. onUpgrade()　　D. getReadableDatabase ()

(7) ContentProvider is one of the four major components of the Android, write a ContentProvider, need to be done in the listing file configuration, configuration < provider > tag, which attribute is required? (　)

A. android: name　　B. android: authorities

C. android: exported　　D. A and B

(8) Read the following program

UriMatcher myUri = new UriMatcher (UriMatcher NO_MATCH);

MyUri. AddURI (" iet. Jxufe. Cn. Will. Myprovider ", "person", 1);

MyUri. AddURI (" iet. Jxufe. Cn. Will. Myprovider ", "the person / #", 2);

Int result = myUri. Match (Uri. Parse (" content: / / iet. Jxufe. Cn. Will the myprovider/ person / 10 "));

After the program execution, the result is (　).

A. 1　　B. 1　　C. 2　　D. 10

(9) The role of the ContentProvider is shared data, and exposure to operation interface,

other applications by ( ) to manipulate the ContentProvider exposed by the data.

A. ContentValues B. ContentResolver

C. URI D. Context

(10) If an application through the ContentProvider Share data that other applications can operate on the data, you can use the query method. to read data. Which kind of object should be used in invoking the query method?( )

A. ContentResolver B. ContentProvider

C. SQLiteDatabase D. SQLiteHelper

(11) When developing Android applications, if you want some structured data in the local store, you can use the database, the small relational database embedded in the Android system is ( ).

A. MySQL B. SQLite C. DB2 D. Sybase

8. Four components of the Android Service and BroadcastReceiver

(1) Which method does not belong to the Service lifecycle callback method?( )

A. onCreate() B. onBind() C. onStart() D. onStop()

(2) Which method can improve the priority of Service?( )

A. setLevel() B. setPriority() C. upgrade() D. startForeground()

(3) On the ServiceConnection interface onServiceConnected() method of the trigger condition description correct is ( ).

A. a successful execution of bindService() method

B. bindService() method performs successfully and onBind() method returns the empty IBinder object

C. after a successful execution of the onCreate() method and onBind() method in Service

D. after a successful starting of the onCreate() and onStartCommand() method in Service

(4) To develop the Service component, we should develop a kind of class inherited the system-provided Service class. Which methods must be implemented?( )

A. onCreate() B. onBind()

C. onStartCommand() D. onUnbind()

(5) Which statement about the service started by startService() and bindService() is not correct? ( )

A. startService() is not associated with the visitor, after running the Service, the bindService() to run the Service will be survival with visitors

B. startService() operation Service will callback onStartCommand() method, and bindService operation Service will callback onBind() method

C. startService () operation of the Service can't communicate with visitors, data transmission, bindService () to run the Service can be communication between

visitors and the Service, the data transmission

D. bindService operation Service must implement the onBind() method, while the startService () operation Service would not have this requirement

(6) Which statement about using AIDL to complete remote Service method invocation is not correct? (　)

A. AIDL interface source code must be defined. AIDL ends with the interface name and AIDL filename can be same also

B. the content of AIDL document is similar to the Java code

C. to create a Service in the Service's onBind() method returns the AIDL interface object

D. AIDL interface and methods cannot add access modifier before the public, private, etc.

(7) Android mobile phones send a broadcast message when they are switched on. If you want to let the application start when booting, you only need to receive in the application of the radio and then start the service. The value of the broadcast Action is (　).

A. Intent. ACTION_BOOT_COMPLETED

B. Intent. ACTION_MAIN

C. Intent. ACTION_PACKAGE_FIRST_LAUNCH

D. Intent. ACTION_POWER_CONNECTED

(8) Which description about the BroadcastReceiver is correct? (　)

A. BroadcastReceivers can only be registered in the manifest file

B. BroadcastReceivers can't cancel after registration

C. BroadcastReceivers can only receive custom broadcast messages

D. BroadcastReceivers can separate registration and cancellation in the Activity

9. Extension

(1) When doing unit test on Android, the configuration should be carried out in the listing file, which statement is wrong? (　)

A. user need to configure instrumentation in < application > tag of AndroidManifest file list

B. user need to configure instrumentation in < manifest > tag of AndroidManifest file list

C. user need to configure users-library in < application > tag of AndroidManifest file list

D. user need to make the test class inherit AndroidTestCase class

(2) The number of levels of Log information in Logcat view is (　).

A. 3　　　B.4　　　C. 5　　　D. 6

(3) Which JSON data is not correct?(　)

A. [" Java ", "android"]

B. [{" Java "}, {} "android"]

C. {" id ": 1," name ":" Java "}

D. [{" id ": 1," name ":" Java "}, {" id ": 2," name ":" android "}]

(4) Before the MediaPlayer broadcast audio and video resources, which method to complete the preparation you need to call? ( )

A. setDataSource() B. prepare() C. begin() D. ready()

(5) Using the MediaPlayer broadcast of music files stored in external memory card (sdcard) operation step is ( ).

A. you can use the MediaPlayer.create() method which deliver the file path to return to the MediaPlayer object, and ready to play

B. direct the path of the music file into the MediaPlayer constructor, and ready to play

C. create MediaPlayer object first, and then call it setDataSource() methods Settings file source, and ready to play

D. Both A and C

(6) The Android VM virtual machine running file's suffix is ( ).

A. class B. apk C. dex D. XML

Appendix C　The Practice of Android Programming

1. Please design and implement interface as shown in the Figure C-1 below.

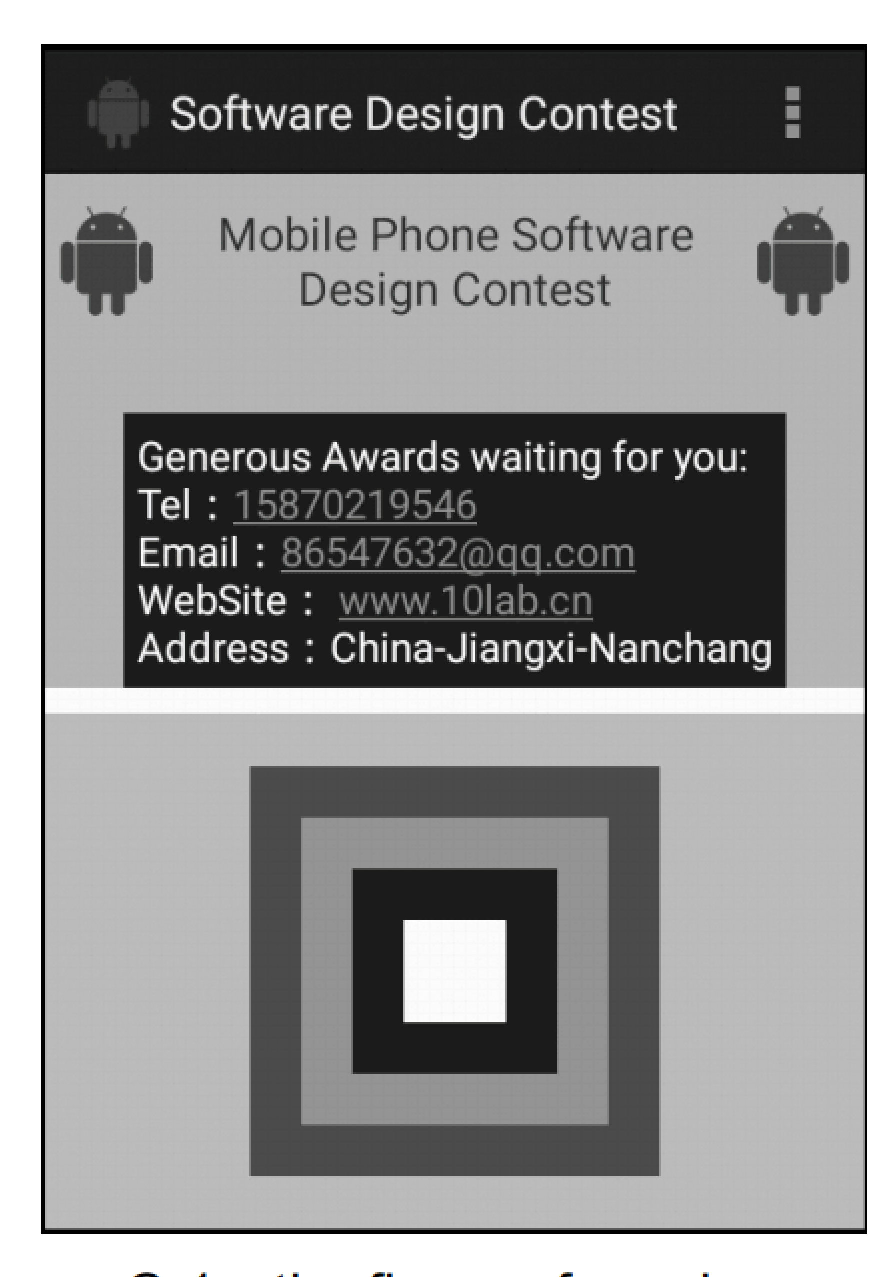


Figure C-1　the figure of running results

Interface requirements:

A. overall using vertical LinearLayout, the screen was divided into fluctuation two parts, top and bottom part for the spacing of 10 dp, on the part of the background color: # aabbcc, the background color of the next part is: # ccbbaa.

B. the upper part contains two TextView controls, the first TextView control is used to display the title information, and in the right and left of the title words each have an icon, the icon for the application icon, centered title words and pictures. On top of the header information and margin is 10 dp, title text is: Mobile phone software Design Contest, the title text size is 18 sp. Second TextView control horizontal center display in the bottom of the upper part, the text content is: Generous Awards waiting for you:\nTel:15870219546\nEmail: 86547632@qq.com \nWebSite：www.10lab.cn\nAddress：China-Jiangxi-Nanchang, text color: white (# FFFFFF), text size: 16 sp, background color: blue (# 0000 ff), the margin: 5 dp, the telephone, E-mail, url text in the content display as the form of links.

C. the lower section contains four TextView, size: 160 * 160, 120 * 120 * 120, 80 * 80, 40 * 40, unit of dp, colors are: red (# ff0000), green (# 00 ff00), blue (# 0000 ff), white (# FFFFFF) and centered.

2. Please design and implement interface as shown in the Figure C-2 below:

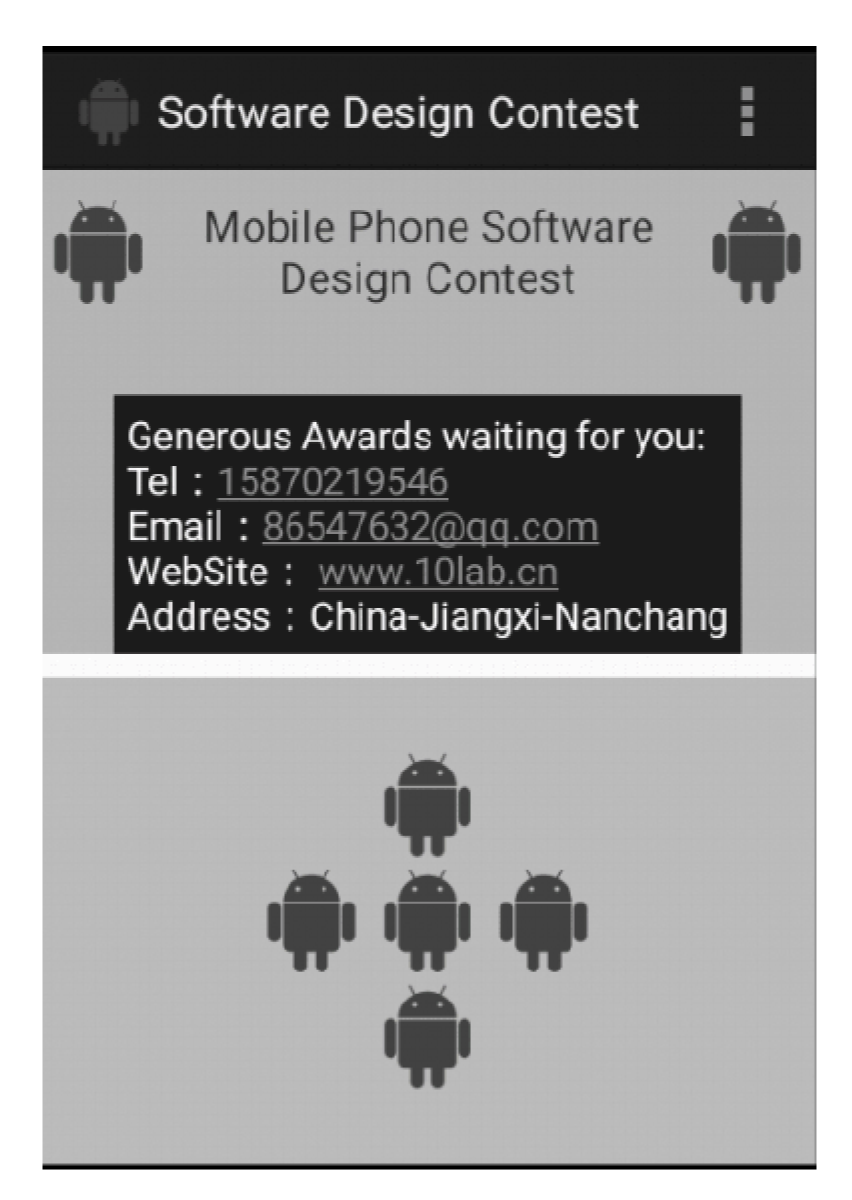


Figure C-2 the figure of the running results

Interface requirements:

A. overall using vertical LinearLayout, the screen was divided into fluctuation two parts, top and bottom part of the spacing of 10 dp, on the part of the background color: # aabbcc, the background color of the next part is: # ccbbaa.

B. the upper part contains two TextView controls, the first TextView control is used to display the title information, in the right and left of the title words each have an icon, the icon is the application icon, centered title words and pictures. On top of the header information and margin is 10 dp, title text is: Mobile Phone Software Design Contest, the title text size is 18 sp. Second TextView control horizontal center display in the bottom of the upper part, the text content is: the big prize is waiting for you to take \ n tel: 15870219546 \ n E-mail: 86547632 @qq.com \ n's official website: www.10lab.cn\n site: China - jiangxi, nanchang, text color: white (# FFFFFF), text size: 16 sp, background color: blue (# 0000 ff), the margin of 5 dp, text content in the telephone, E-mail, url in the form of links to display.

C. the lower part contains five ImageView, five ImageView shown in the pictures are for the application icon, centered on an ImageView, the other four ImageView in it is directly above, below, left, right, as a whole is in the middle of the second half.

3. Please design and implement interface as shown in the Figure C-3 below:

Interface requirements:

A. interface contains two widgets TextView and ListView. TextView for display title information, title says: NanChang attractions introduction, text size: 24 sp, background color: # ccbbaa, alignment for middle and margin for 10 dp.

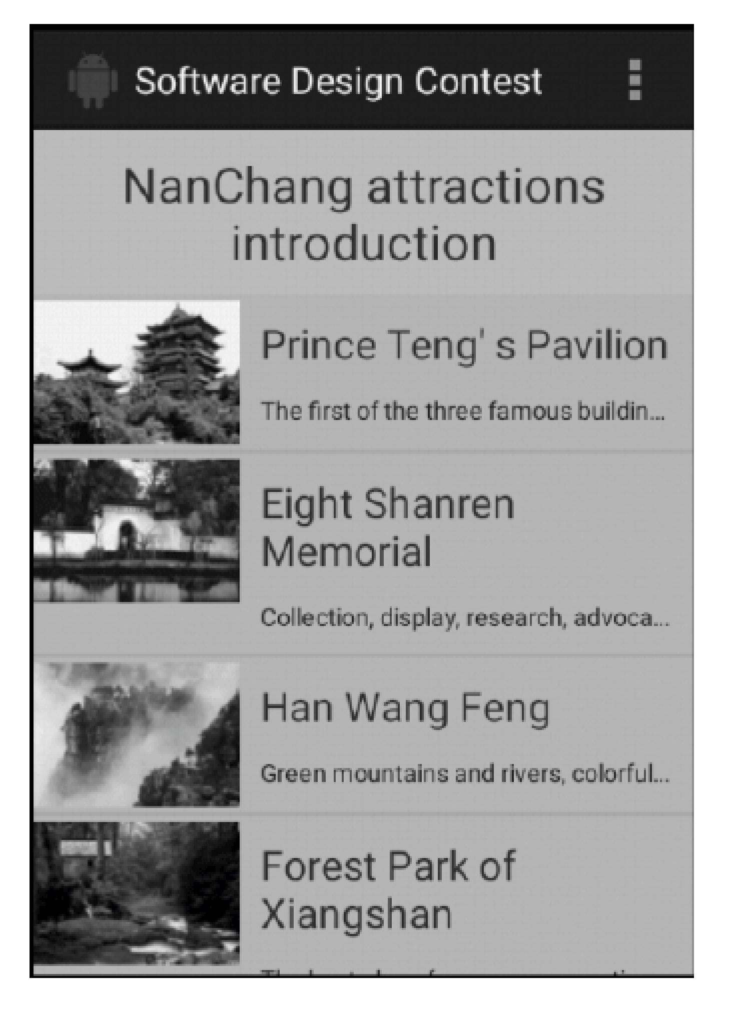


Figure C-3　the figure of the running results

B. ListView display all attractions information, and represents a site.Each scenic spot contains three parts information, scenic spot images, attractions name, and scenic spot introduction. The background color of the ListView is # aabbcc, items with the divider between the sizes of 2 dp, color to gray (# aaaaaa).

C. in each item of ListView contains three controls: a ImageView is used to display attractions images, two TextView display name of the attractions, scenic spots introduction. The ImageView size: 100 * 75, scenic spot name text size for: 20 sp, scenic spot introduction text size: 12 sp, color: # 0000, single row shows that when the content more than width, omit the back of the text, to replace. Had the images and relevant text introduces BA3 folder.

4. Realize image browsing, application effect as shown in the Figure C-4 below:

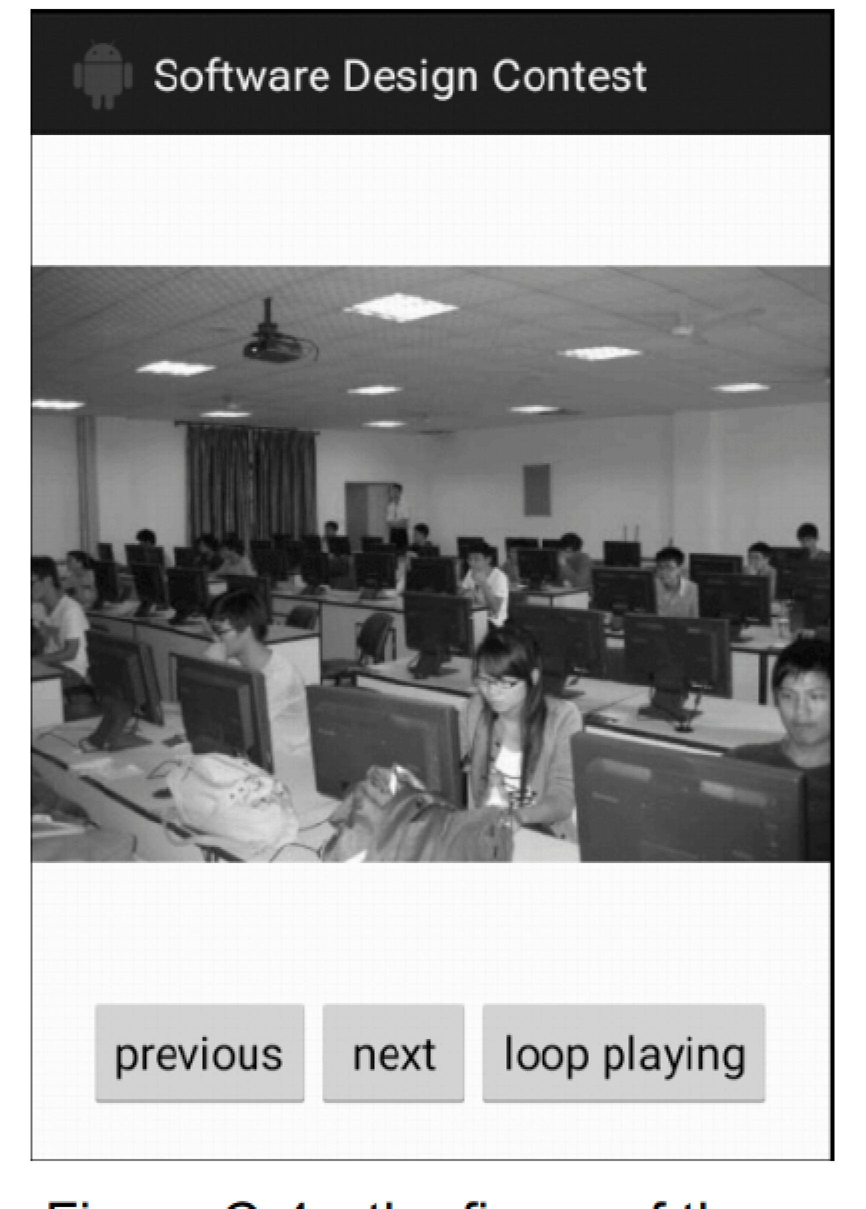




Figure C-4　the figure of the running results with different operations

Interface requirements:

Interface contains an ImageView and three Buttons, the default display ImageView is the first picture file1.jpg. Three Buttons, laied out in horizontal center, button labels respectively

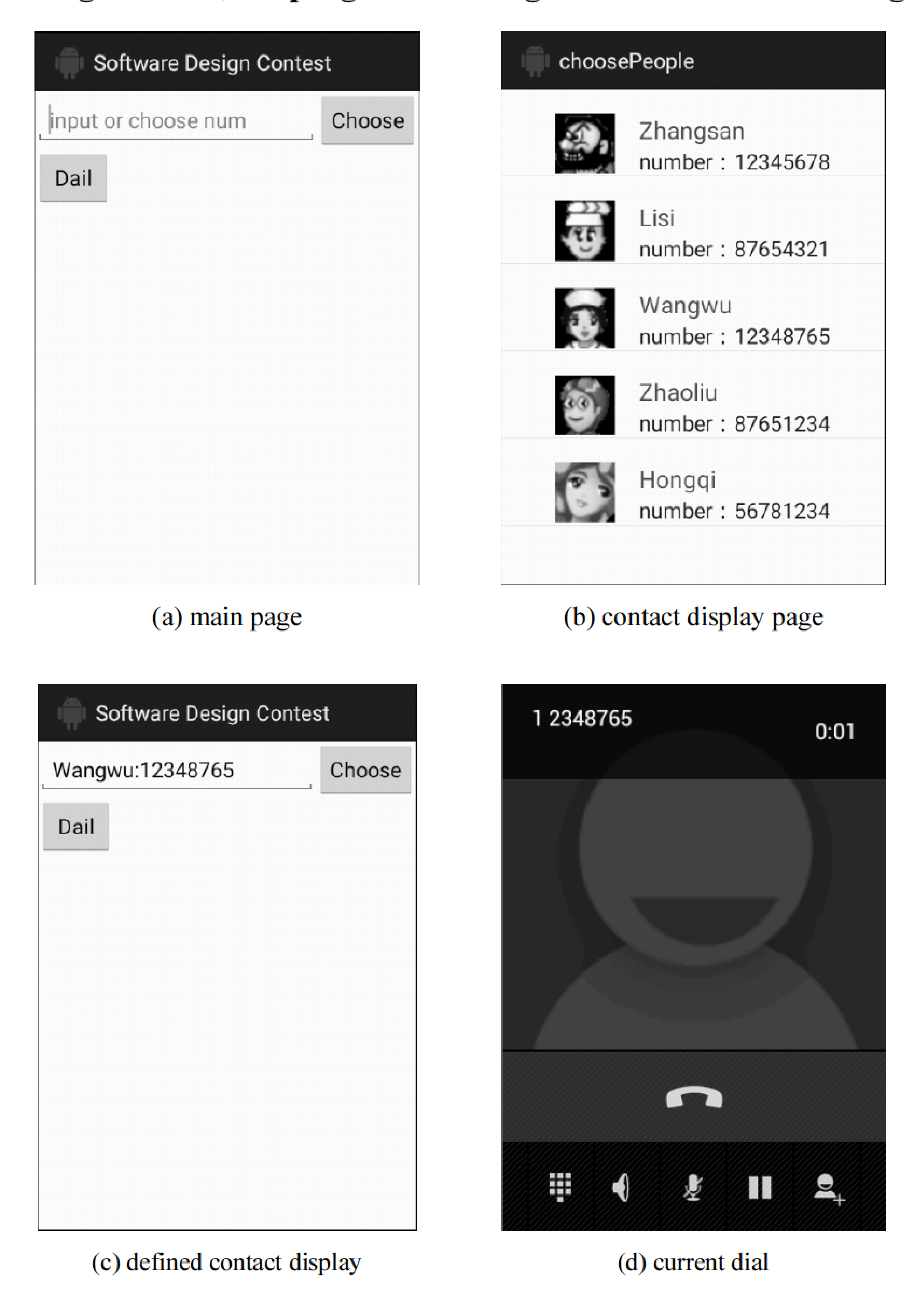
on the “provious”, “next”, “loop playing”.

Functional requirements:

A. as “previous”, “next” button to add the event listener, after click the button to switch to the proevious picture or the next, event handling any method, can use to bind to tag can also use the listener;

B. for a “loop playing” button to add the event processing, after clicking this button, program can automatically cycle between multiple pictures showing every 2 seconds to switch, and button text changes, according to “stop looping”, after click again to stop looping, button to display a “loop playing”. Related images resources have on BC1 folder. (Key: button in the loop switch between play and stop playback, click the button to play and stop cycles).

5. Realize dialing function, the program running effect as shown in the Figure C-5 below:



(a) main page

(b) contact display page

(c) defined contact display

(d) current dial

Figure C-5 the interface of application running

After running the program, click the button “choose” person in Figure C-5(a), the

windows will jump to the contact list page in Figure C-5(b), select one from the list, It will return to the home page and display the contacts user selected in Figure C-5(c), click “Dial” button, it will call dialing function, then system begin to dial.

Interface requirements:

Home page contains a EditText, two Buttons, the EditText, and Button “choose” horizontally, the width of the EditText is the width of the screen minus the width of the button.

Select the contact page contains a list of ListView control. Each item in the list contains three parts information, icon, name, phone number. The icon size of 50 * 50, name text size: 20 sp, color red (# ff0000), telephone number text size: 18 sp, color is blue (# 0000 ff).

Functional requirements:

A. after clicking the “choose” Button, jump to the contact list page.

B. select one of the contact list, contact name and number will be returned to the main page. And displayed clicking the text in the EditText, pay attention to display content is: name: number.

C. after clicking the dial button can invoke the system function, to dial out.

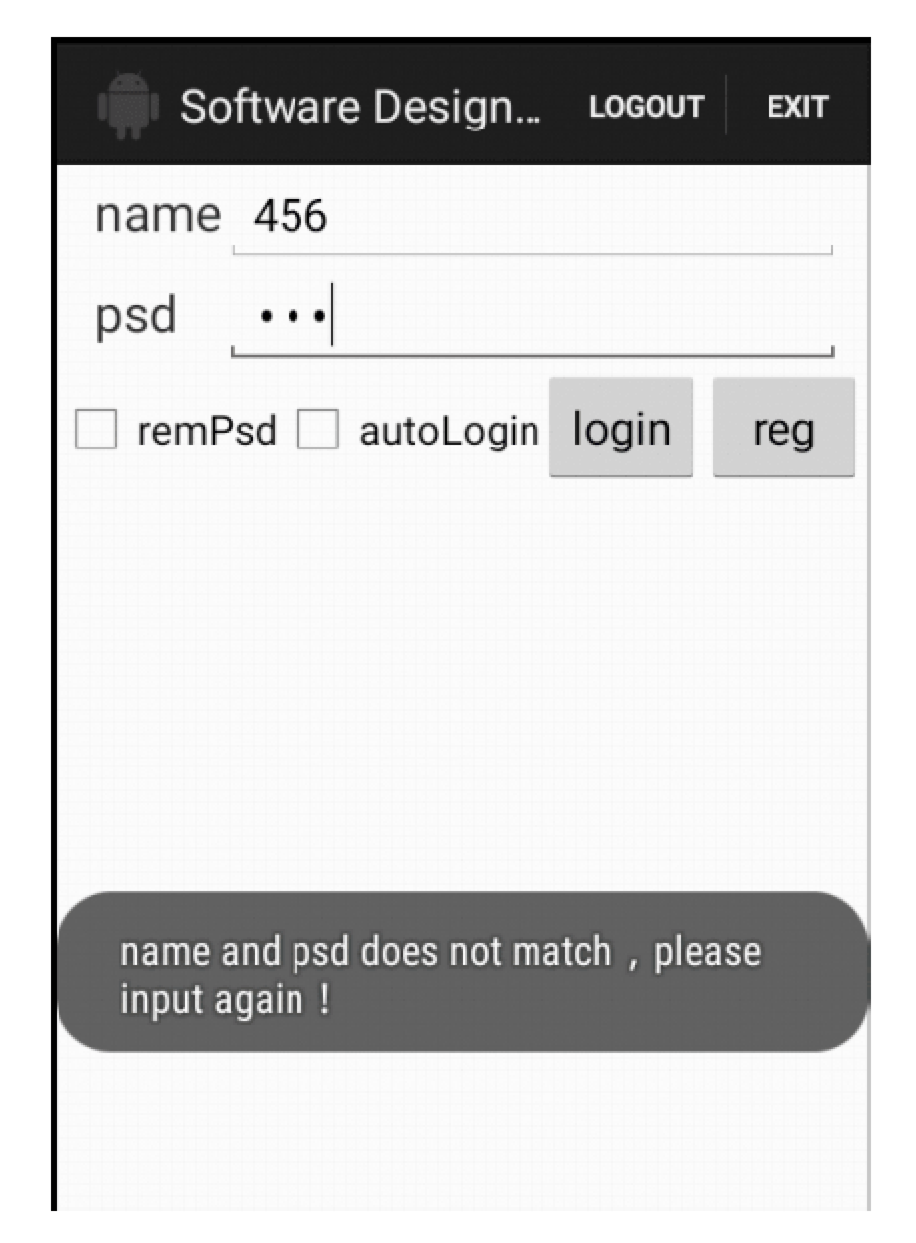
(Tips: dial-up function call system shall provide the relevant permissions, call the permissions for: <uses-permission android:name="android.permission.CALL_PHONE"/>. System calls action as: Intent. ACTION_CALL)

Related images resources have on BC2 folder.

6. Realize the register, login, programs run effect as shown in the Figure C-6 below:

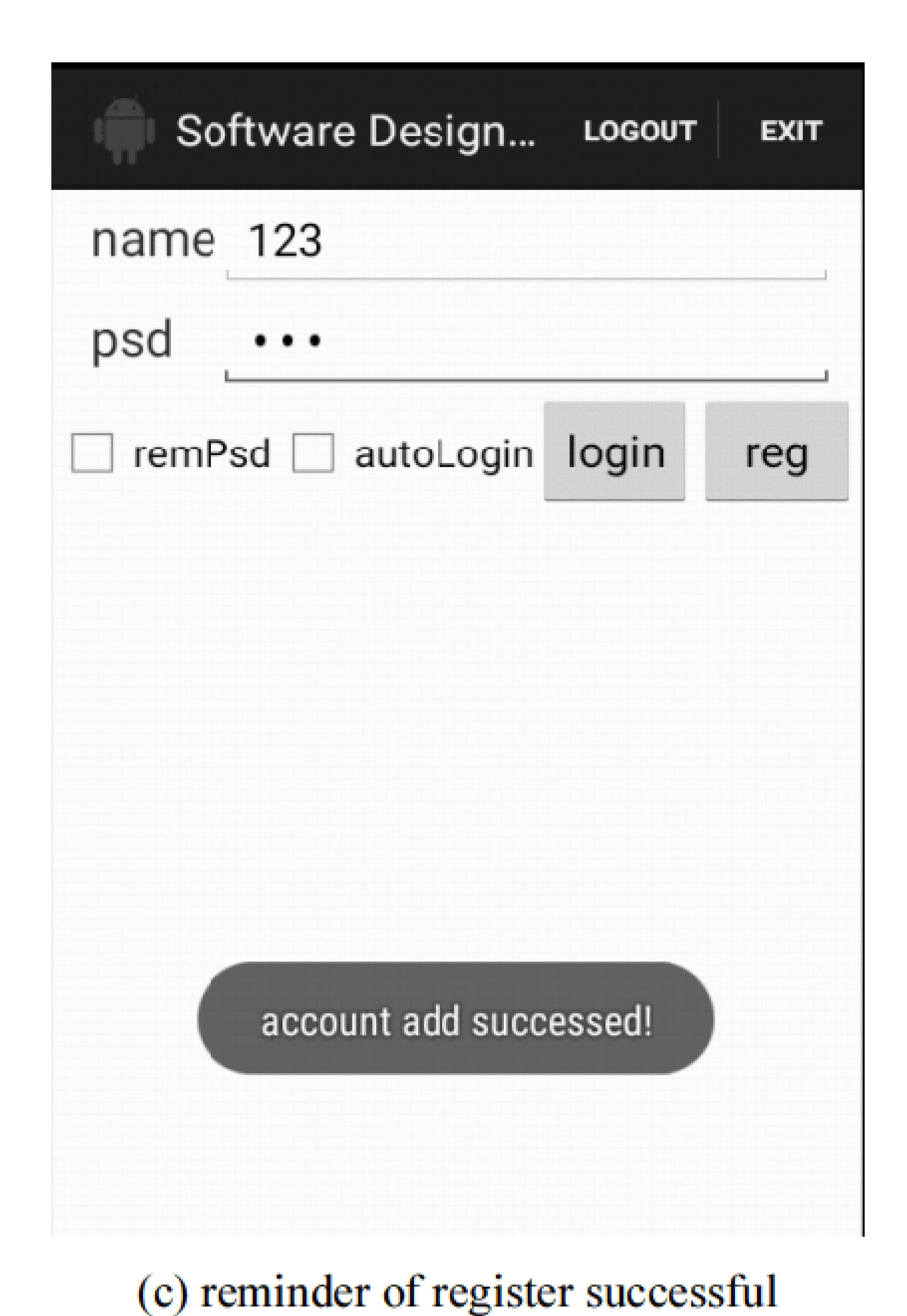


(a) main page

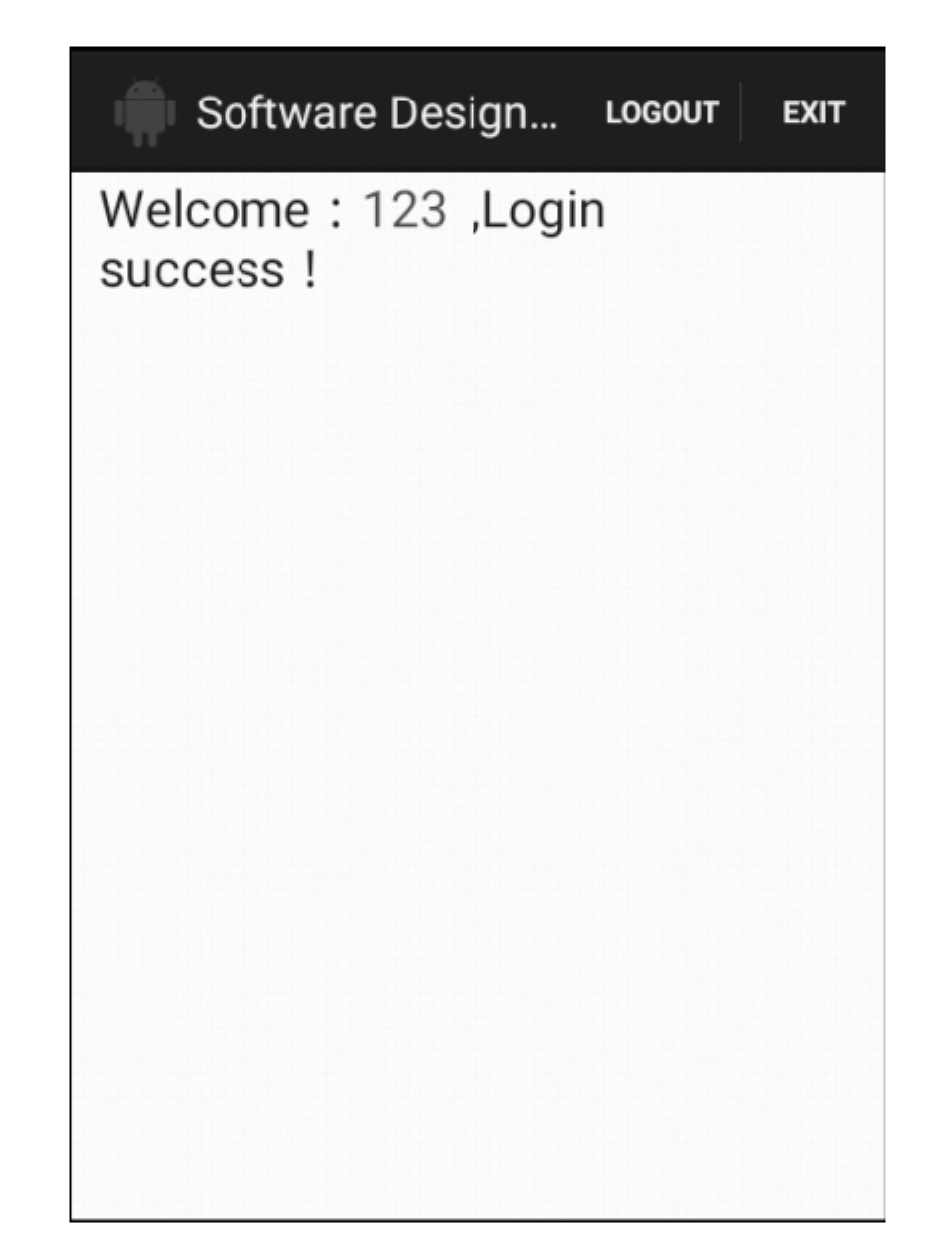


(b) reminder when user name or password incorrect

Figure C-6　some interface when the application is running



(c) reminder of register successful



(d) display of loading successful

Figure C-6 (continued)

Program runs the main interface as shown in Figure C-6(c), initialization, any input account and password can not login prompt the user name and password is not correct, need to register first and then you can log in. After entering account and password, clicking the register button, a user can be deposited with the database records, and then input the account and password, from the database query found the account, you can log in, if the database does not exist the account, the login prompt failure information.

Users can save your login information, including the remember password and automatic login, check the remember password, do not need to enter the next time you login account and password, check the automatic login after the jump straight to the next time you log in the welcome screen.

Interface requirements:

This program contains two pages, one is the login/registration home page, a login is successful according to user information page, the two page has been provided, under the BC3 folder provides a working an incomplete application, import it into Eclipse, on the basis of perfecting related functions.

Functional requirements:

A. a program is running, from the login. The XML file for the user to save the related information, if the user had selected automatic login, displayed a welcome to the login page directly, or show the login page, and then determine whether to remember the password, if remember password in the appropriate EditText shows the last input user name and password.

B. to complete the login button event handling, click the login button, the first to determine whether there is entered by the user in the database user name and password, if there is no hint failed login information, if there is a switch to the welcome screen, show the login

successful information, at the same time save users remember password and automatic login information to login.

C. the complete registration button event handling, click the register button, the user input user name and password saved into the database, and to prompt registration information.

D. complete menu item selected event handling, select the logout menu item, cancel the automatic login page to switch to the main interface, select the exit when a menu item, exit the application.

7. To control the progress function, application effect as shown in the Figure C-7 below:

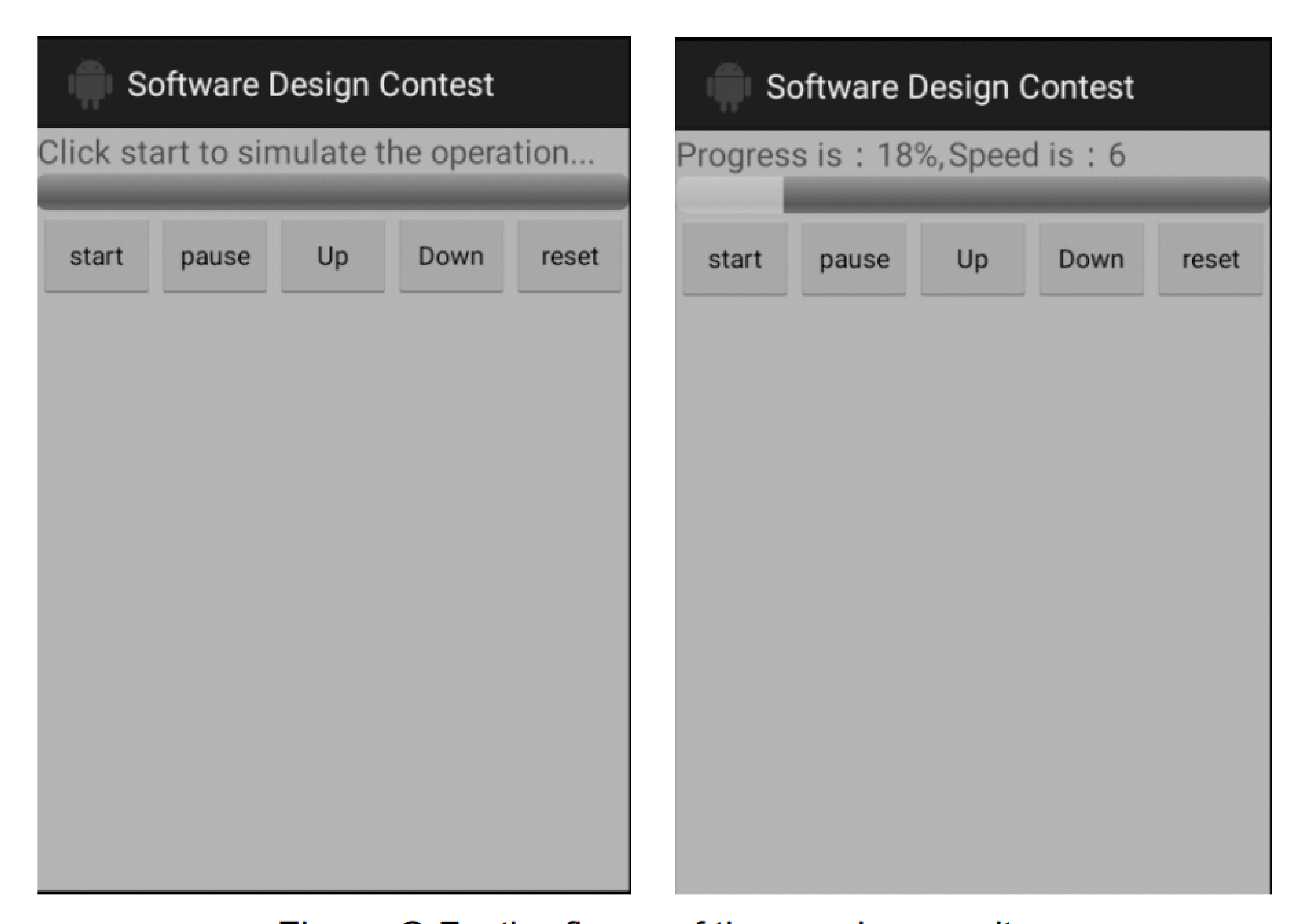


Figure C-7　the figure of the running results

Interface requirements:

Interface design has been implemented in BC3 folder provides an incomplete application; Then you can import it into Eclipse, on the basis of perfecting related functions. Run the program directly, will be shown at left.

Functional requirements:

A. perfect event handling methods, perfecting the start button, click the start button, the progress bar began to change, in the progress bar at the top of the text, real-time display the progress of the information, including the percentage of the current schedule execution, increasing the speed of progress.

B. perfect event handling methods, perfecting the pause button, click the pause button, the interface is not change, keep to the previous state, at this time in addition to click the pause button can perform related operations, progress can be change.

C. perfect event handling methods, improve the speed button, click the accelerate button, in the original speed above 5, according to the current speed is changing.

D. perfect event handling methods, improve the speed button, click the speed button, on the speed of the original subtract 5, but the lowest rate of 1, according to the current speed is changing.

E. perfect event handling methods, perfecting the reset button, click the reset button, the interface back to the initial state.

F. on the basis of the original program can completely own implementation, achieve function.

Appendix D The Informal Test of Android Programming

Part I Single option (2 point per question, 20 points in total)

(1) The command of launching the Android SDK and AVD manager is ( ).

A. adb B. avd C. android D.emulator

(2) Assuming that the phone's screen width is 400 px, we use horizontal LinearLayout placed five Buttons in layout, set the width of each button is 100 px, then the program runs, the interface display effect of ( ).

A. Automatically adds the horizontal scroll bar, drag the scroll bar to view the five Buttons

B. Can only see the four Buttons, beyond the screen width part cannot display

C. Button automatically narrow width can see five Buttons

D. Program is run error, unable to display

(3) Which attributes value can only be true or false in relative layout? ( )

A.android: layout_below B. android: layout_alignParentLeft

C.android: layout_alignBottom D. android: layout_toRightOf

(4) The right way to set a listed as a retractable column in the table layout is ( ).

A. Set the TableLayout properties: android: stretchColumns = "x", x represents the serial number of the column

B. set the TableLayout properties: android: shrinkColumns = "x", x represents the serial number of the column

C. Set specific properties: android: stretchable = "true"

D. Set the concrete column attributes: android: shrinkable = "true"

(5) Which option of the color value is illegal?( )

A. # aaa B. # bbbb C. # ccccc D. # dddddd

(6) When using asynchronous task processing, the following method that can't change the interface control state is ( ).

A. onPreExecute() B. doInBackground ()

C. onPostExecute () D. onProgressUpdate ()

(7) Which method does not exist in the life cycle of Activity? ( )

A. onStart () B. onCreate () C. onPause () D. onFinish ()

(8) Which option is indispensable when configuring the Activity? ( )

A. android: name B. android: icon C. android: label D. < intent - filter... / >

(9) SharedPreferences data is saved on the phone by ( ) format.

A. XML B. TXT C. json D. user defined

(10) through openFileOutput (String name, int mode) reads a file on your mobile phone, if the second parameter value of 3, said the file ( ).

A. Is a private data, can only be accessed by application itself

B. Can be read by other applications

C. Can be written by other applications

D. Both can be read by other applications and can be written to other applications

Part II True or False (2 point per question, 10 points in total)

1. Strings.xml in Android applications can only saves some information of strings constants. ( )

2. The assets in the Android project resource files in the directory can be accessed by R resource list. ( )

3. An Intent object can only contain one Action attribute, but can contain more than one Category attributes. ( )

4. The development of the context Menu, need to rewrite the Activity onCreateOptionsMenu (Menu menu) method, if you want to make application responding to the click event of Menu items, you still need to rewrite the Activity's onOptionsItemSelected (MenuItem mi) method. ( )

5. When registering ContentProvider component, you must specify the android: authorities attribute's value. ( )

Part III Operation (20 point per question, 20 points in total)

Create an Android application, application can run normally, the running effect is shown in Figure D-1.

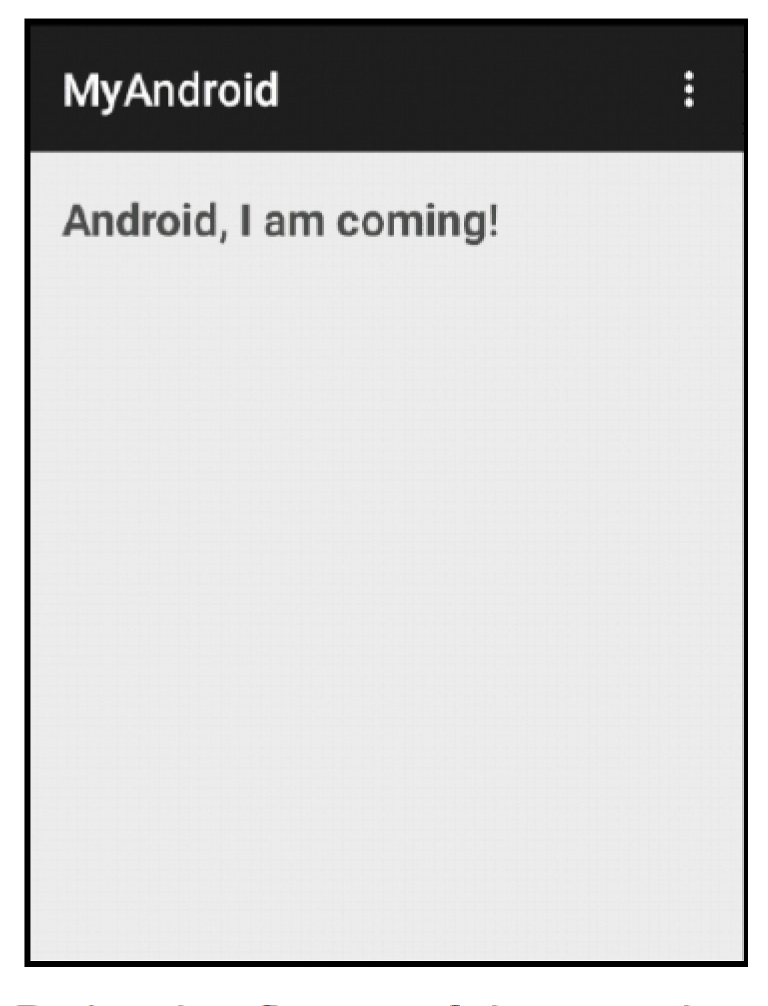


Figure D-1 the figure of the running results

A. program icon for a given image, the picture is 001.png in 001 folder;

B. entitled “MyAndroid”, display text for the “Android, here I come!” size: 20 sp, color: red;

C. created a completed description of the entire application process (similar to the experiment report) through capture.

Part IV Basic programming (20 point per question, 40 points in total)

1. Loading LayoutTest projects of 002 folders into Eclipse, and running, as shown in Figure D-2 results, on the basis of this program, please revise and supplement, and make it run as shown in Figure D-3. All these in Figure D-2 and Figure D-3.The images needed are under the 002 folder.

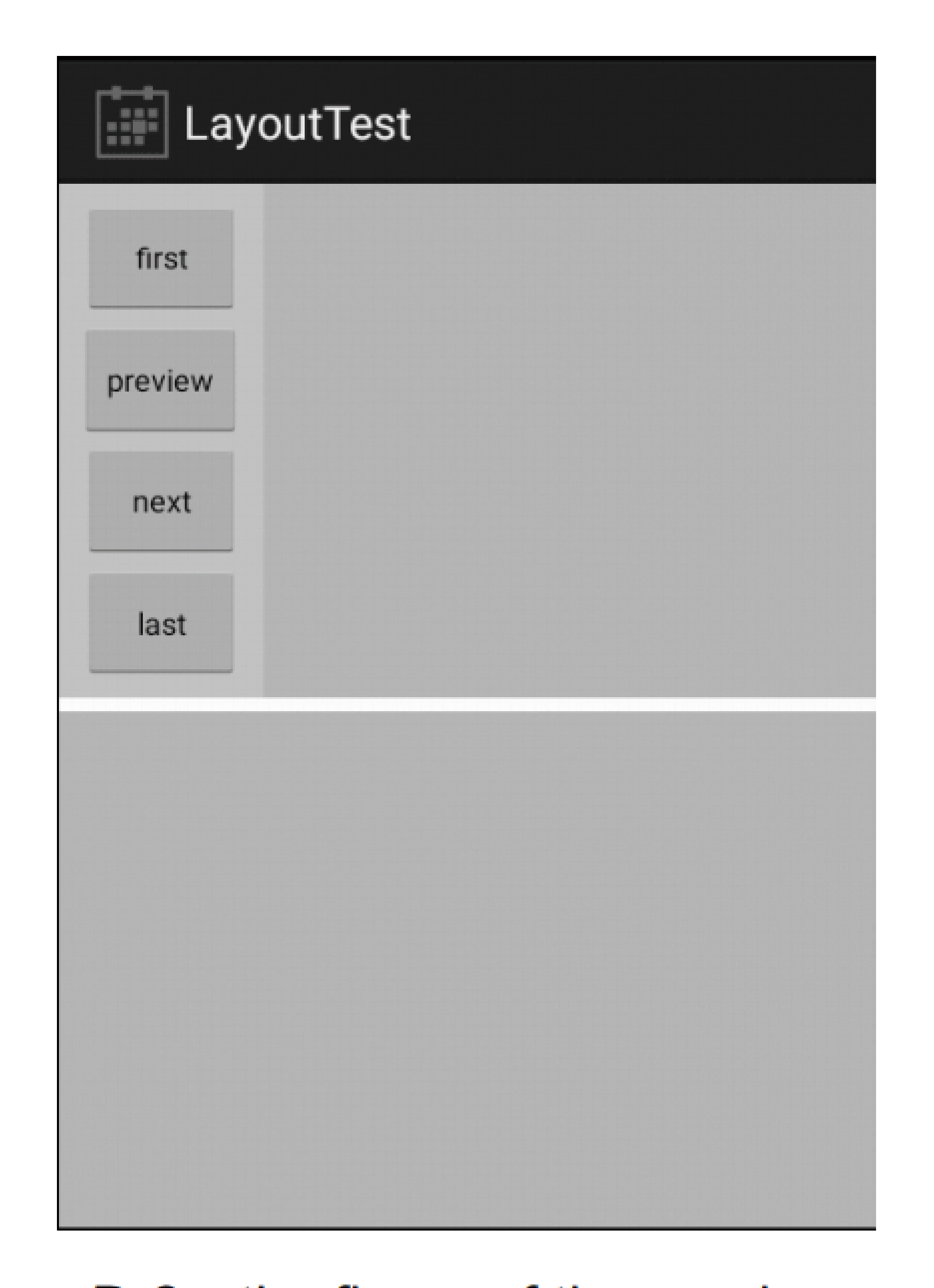


Figure D-2 the figure of the running results

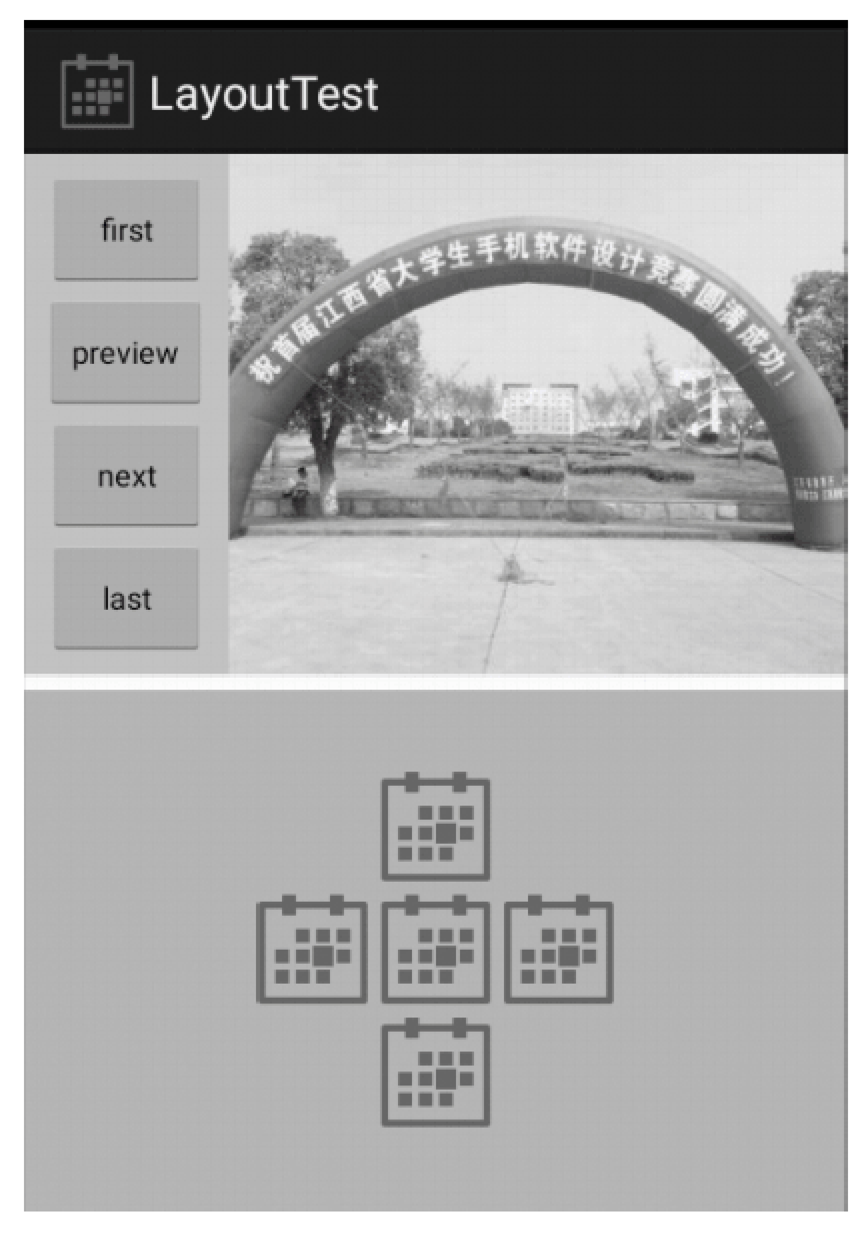


Figure D-3 the figure of the running results

A. “first”, “previous” and “next” and “last” button to the overall level of centered in the parent container;

B. ImageView according to the given picture jingsai. The JPG, required zooming aspect ratio doesn’t change, and completely covered ImageView;

C. In the next half of the area to place 5 ImageViews, every ImageView image resources for a given leaf.png. On one of the ImageView as center, others at the left, above, to the right, below, and the overall placed in middle of the container.

2. Loading ImageTest project in 003 folders into Eclipse and running, as shown in Figure D-4, on the basis of this program, please revise and supplement, in order to realize the following functions.

A. as “previous”, “next” button to add the event processing, after clicking the button to switch to the previous or the next, requires the use of event listeners to implement;

B. for a “loop playing” button to add the event processing, requires the use of binding to label the mechanism of implementation, after clicking this button, program can automatically cycle between multiple pictures show, every 2 seconds to switch.

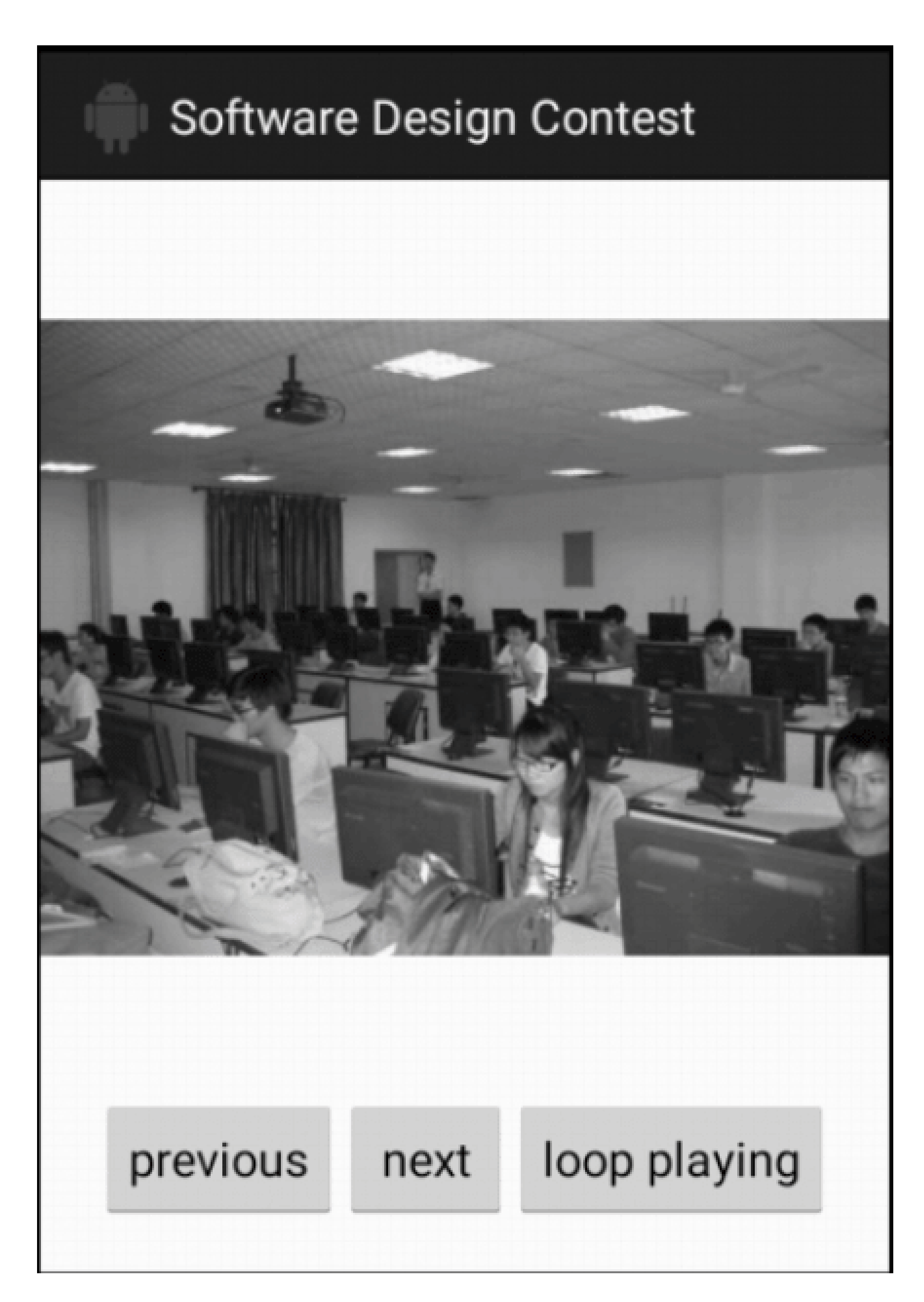


Figure D-4 the figure of the running results

Part V Comprehensive programming (10 point per question, 10 points in total)

Choose a topic from the following two questions (A or B) for programming. And tag your chosen topic in the paper.

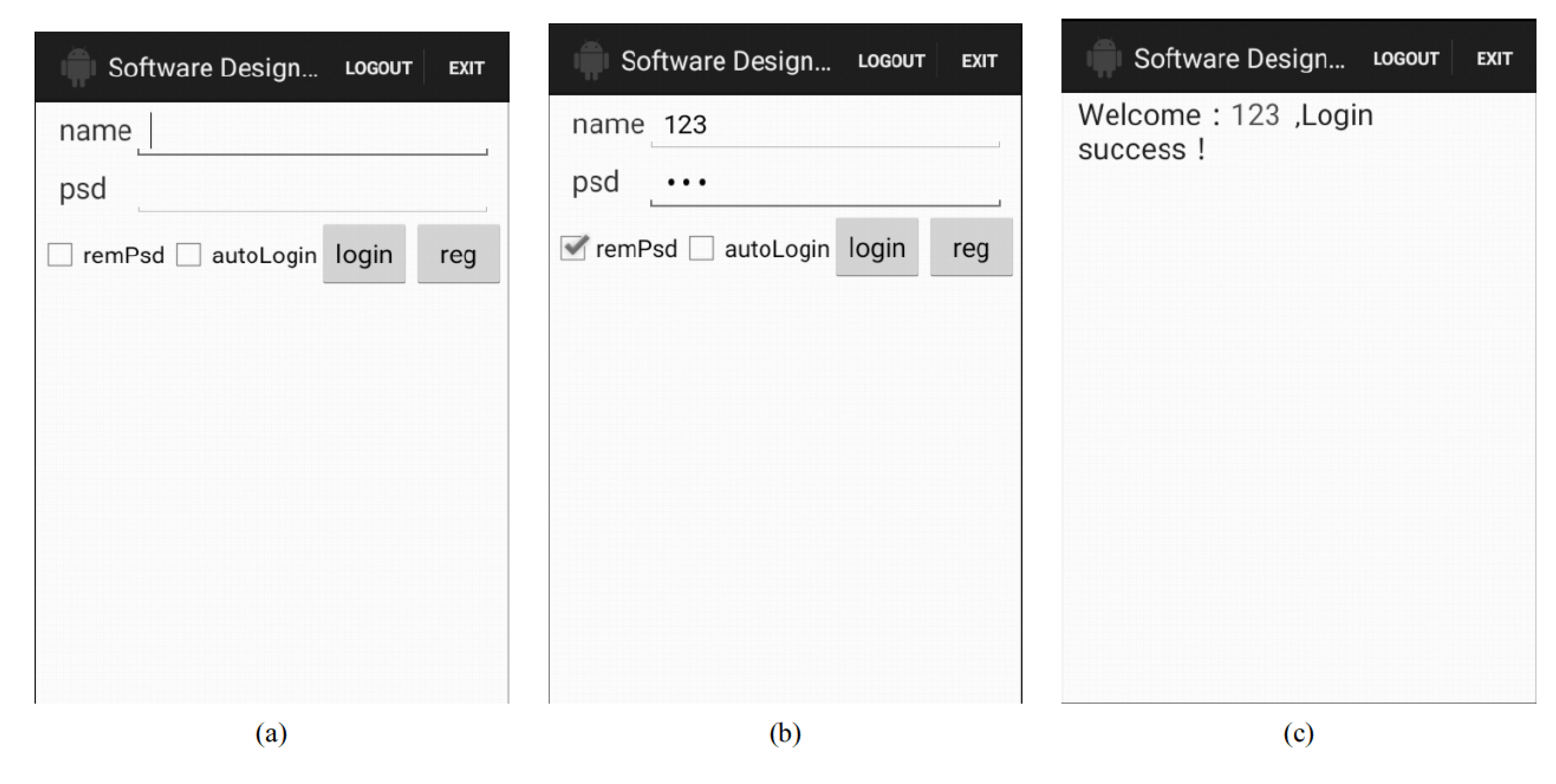
A. Please import DataTest project of 004 folders in Eclipse and running, as shown in figure 5 results, on the basis of this program, please revise and supplement. In order to realize the following functions:

a. improve the event processing of the “login” button, after clicking the button. Firstly, query database if there is the account and password, if any, to save the information into SharedPreference, then show the login successful information, if there is no criterion by Toast show related information;

b. perfecting event handling of the "register" button, click the button, insert a user record into the database;

c. as a "logout" and "exit" menu item to add event processing, after click “logout”, to show the login interface, click the "exit", close the current Activity;

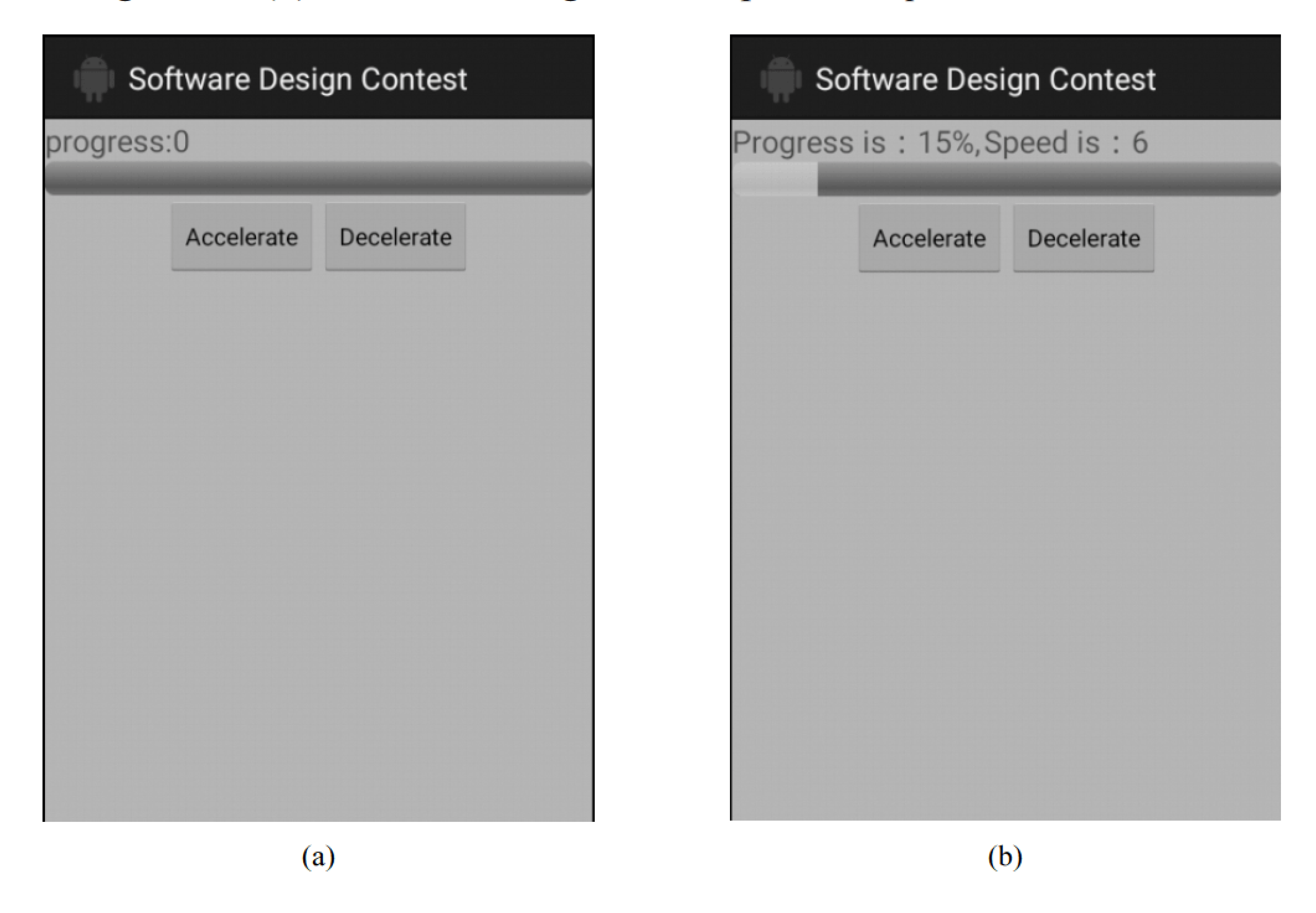
d. program is running, SharedPreference will be saved in the information displayed in the corresponding controls, and for example, to remember the password, the running effect is shown in Figure D-5(B), automatic login, after running effect as shown in Figure D-5(C). all these in Figure D-5.



(a) (b) (c)

Figure D-5 the figure of the running results with different operations

B. Import ServiceTest project of folder005 into Eclipse and running, as shown in Figure D-6(a), please revise and supplement on the basis of this program, eventually reach the effect as shown in Figure D-6(b). All these in Figure D-6.Specific requirements are as follows:



(a) (b)

Figure D-6 the figure of the running results at different times

a. starting service in the MainActivity back-end which is used to count;

b. perfecting the code of"Accelerate" and "Decelerate" button event handling and respectively add 5 and minus 5 in the original speed,then through the broadcast inform back-end services;

c. receiving broadcast in back-end, changing the speed of counting, sending broadcast regularly, transfering the current value, changing progress bar display according to the numerical after receiving broadcast in front.The text dynamically displays the current progress bar at the top of progress and the speed of the current count, the progress bar to change according to the current count.